

# **Projet ToDo & Co**

Audit de qualité et de performance

04/04/24

# 1 Sommaire

## Table des matières

1 Sommaire.....	2
2 Introduction.....	3
2.1 Le projet original.....	3
2.1.1 Failles de sécurité.....	3
2.1.2 Code non maintenable.....	3
2.1.3 Pas de Poo.....	4
2.1.4 Symfony non maintenu.....	4
2.1.5 Php non maintenu.....	4
2.1.6 Audit de qualité.....	4
2.1.6.1 Sévérité des erreurs.....	5
2.1.7 Audit de performance.....	6
2.1.7.1 /login (post).....	6
2.1.7.2 /tasks (get).....	6
2.1.7.3 /task/id/toggle.....	7
2.1.7.4 /task/create (post).....	7
2.1.7.5 /task{id}/edit (post).....	7
2.1.7.6 /task{id}/delete (get).....	7
2.1.7.7 Résumé.....	7
2.2 Plan de remise en route.....	8
2.2.1 Option 1 : upgrader les versions.....	8
2.2.2 Option 2 : Refonte (From Scratch).....	8
2.2.3 Motivation pour la solution adoptée.....	8
3 Nouvelle version V2.....	9
3.1 Le projet modifié.....	10
3.1.1 Audit de qualité.....	10
3.1.1.1 Sévérité des erreurs.....	10
3.1.2 Audit de performance.....	13
3.1.2.1 /login (post).....	13
3.1.2.2 /tasks (get).....	13
3.1.2.3 /task/id/toggle.....	13
3.1.2.4 /task/create (post).....	13
3.1.2.5 /task{id}/edit (post).....	13
3.1.2.6 /task{id}/delete (get).....	14
3.1.2.7 Résumé.....	14
4 Synthèse.....	14
4.1 Mesures comparative de la qualité.....	14
4.2 Mesures comparative des performances.....	15
5 Conclusion.....	16

## 2 Introduction

Le projet ToDo & Co consiste en un logiciel de gestion des tâches, que les employés peuvent ajouter, déclarer comme remplies ou supprimer.

Il a fallu dix ans (estimation) pour débloquer les fonds permettant de lancer véritablement le projet, qui dans sa version préliminaire posait les bases de son fonctionnement.

Le domaine informatique est encore effervescent et il est impensable qu'un projet conçu il y a si longtemps puisse rester en état de fonctionnement sans une constante maintenance.

Le motif de l'intervention a consisté à remettre le logiciel sur les rails d'une maintenance applicative et de permettre l'ajout graduel de nouvelles fonctionnalités.

Pour cela il aura fallu commettre une résurrection du projet depuis des cartons poussiéreux et le porter à un stade où, tout en ayant les mêmes fonctionnalités (plus quelques autres qui sont élémentaires) il deviendra exploitable.

Dans sa version originale, les fonctionnalités sont très vagues, et mal déterminées. N'importe qui peut ajouter ou supprimer des tâches quelconques, et il n'y a pas de suivi des auteurs de ces actions.

Le présent document se limitera à dessiner les contours techniques du logiciel, avant et après l'intervention, en effectuant des mesures d'ordre qualitatives et de performance, et ensuite de comparer ces mesures entre la version V1 et v2.

Version 1		Zip source
Version 1,1	« v1 »	Site rendu opérationnel, utilisé pour les tests
Version 2	« v2 »	Refonte From Scratch

### 2.1 Le projet original

Le projet original est une application Symfony 3, sorti en 2015 (source : <https://fr.wikipedia.org/wiki/Symfony>) qui, au plus, tourne sous Php-5.5.9, sorti en 2014 (source <https://www.php.net/releases/>).

#### 2.1.1 Failles de sécurité

Les besoins en sécurité n'étaient pas les mêmes à cette époque, où notamment les serveurs n'étaient sécurisés que dans le cadre de paiements. Aujourd'hui des bots peuvent facilement se frayer un chemin dans les failles de sécurité et garder un moyen de corrompre les données.

La principale source de problèmes vient du firewall, qui à l'époque nécessitait un investissement supplémentaire. Une vulnérabilité majeure a été apportée à la version 3 de Symfony en 2018 sur les redirections après un login, ou sur l'accès direct au chemin des fichiers téléchargés (source <https://www.datasecuritybreach.fr/failles-et-bugs-corriges-dans-symfony-3-4-20/>).

### **2.1.2 Code non maintenable**

La vie d'un projet consiste à maintenir et ajouter des fonctionnalités sans que celles-ci n'occasionne à son tour de gros besoins en maintenance. Pour cela, la façon dont le code est formulé doit être compatible avec les modules qui pourraient être ajoutés et surtout compréhensibles sans avoir à faire des fouilles archéologiques.

### **2.1.3 Pas de Poo**

Un des motifs pertinents auxquels s'attacher est la propension du code à être réutilisable sous forme de services et à bien détacher les rôles. Ce n'était pas le cas dans la version préliminaire, qui semblait écrite à la manière dont le code se présente dans la documentation Symfony pour des raisons de concision.

### **2.1.4 Symfony non maintenu**

Pour passer de la version 3 à la version 7 (actuelle) il faudrait réitérer toutes les évolutions successives une à une, et à chaque fois déboguer ce qui aura été perdu dans la bataille, comme cela arrive souvent.

### **2.1.5 Php non maintenu**

Le langage Php a énormément progressé entre la version 5.5.9 et la version 8.3.4, actuelle. Sans cesse accusé d'être un langage voué à disparaître, depuis ses débuts, il reste le langage le plus utilisé pour le web aujourd'hui, avec un taux de pénétration de 75 % (source <https://kinsta.com/fr/blog/php-est-il-mort/>).

Les versions en-dessous de Php-7.4, sorti en 2019 (source <https://www.php.net/releases/>) ne sont plus prises en charge. Cependant 70 % des entreprises utilisent encore ces versions tant il est coûteux d'affronter des – parfois – dizaines de milliers d'erreurs notices qu'affiche une version supérieure à 8.0, sans parler de tous les tests à refaire (source <https://kinsta.com/fr/blog/versions-php/>).

Les raisons de passer à la version courante (8,3,4) sont l'implémentation de nouvelles fonctionnalités, une sécurité accrue, et une vitesse d'exécution très largement meilleure (source <https://kinsta.com/fr/blog/versions-php/>).

### **2.1.6 Audit de qualité**

Pour le logiciel qui nous concerne, l'audit de qualité de Codacy fournit les appréciations suivantes :

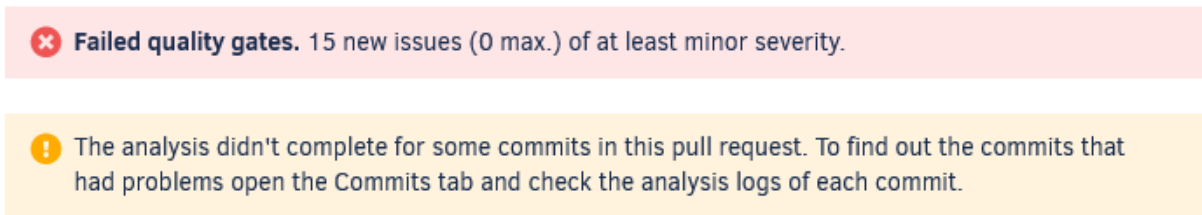


Figure 1: Echec aux tests de qualité

### 2.1.6.1 Sévérité des erreurs

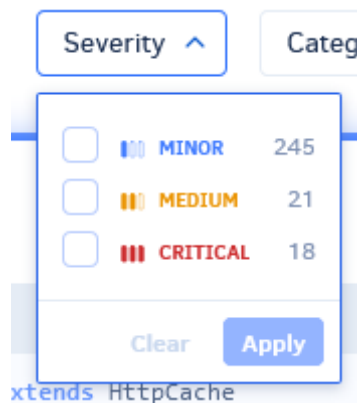
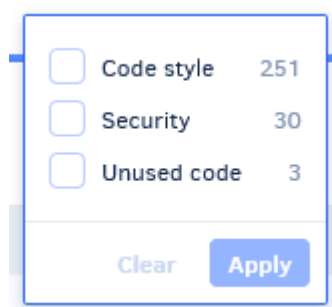


Figure 2: Sévérité des erreurs

Le rapport fait état de **18** erreurs dites critiques :

minor	245
medium	21
critical	18

Les erreurs Medium et mineures n'ont pas une grande influence, elles concernent surtout la qualité du code, mais les erreurs critiques peuvent impacter durablement la vie et la réputation du logiciel.



<input type="checkbox"/>	Code style	251
<input type="checkbox"/>	Security	30
<input type="checkbox"/>	Unused code	3

Clear Apply

Figure 3: Catégories d'erreurs

On peut voir qu'au total, **30** erreurs portent sur la sécurité.

### 2.1.6.2 Quelques exemples d'erreurs critiques

1. « Toutes les sorties doivent être passées par une fonction d'échappement (voir les sections sur la sécurité dans les Manuels du développeur de WordPress), trouvée 'basename'. »

**CRITICAL** | Security XSS

All output should be run through an escaping function (see the Security sections in the WordPress Developer Handbooks), found 'basename'.

web/app\_dev.php

```
16     } {
17     header('HTTP/1.0 403 Forbidden');
18     exit('You are not allowed to access this file. Check '.basename(__FILE__).' for more information.');
```

Introduced by agence2dav, 17 days ago | Time to fix: 5 minutes

**Why is this an issue?**

Security: Escape Output

Related code pattern: Security: Escape Output by PHP\_CodeSniffer

Bien que le WordPress Developer Handbooks n'en ait pas l'air, il est une référence mondiale en terme de hacking (lien <https://developer.wordpress.org/themes/advanced-topics/security/>). Il était le premier à avoir révélé l'ampleur et la complexité inouïe de l'attaque SolarWinds. Et bien que l'erreur signalée paraisse prosaïque (rappelons que SolarWinds a juste changé un numéro de version sur un serveur Ftp pour déclencher une mise à jour généralisée), elle fait partie des failles XSS, qui peuvent être exploitées lors d'une attaque (lien <https://www.vaadata.com/blog/fr/failles-xss-principes-types-dattaques-exploitations-et-bonnes-pratiques-securite/>).

2. « Utilisation détectée d'un index de tableau superglobal possiblement indéfini : `$_SERVER['REMOTE_ADDR']`. Utilisez `isset()` ou `empty()` pour vérifier que l'index existe avant de l'utiliser. »

CRITICAL

Security Input Validation

Detected usage of a possibly undefined superglobal array index: `$_SERVER['REMOTE_ADDR']`. Use `isset()` or `empty()` to check the index exists before using it

web/app\_dev.php

```
13  if (isset($_SERVER['HTTP_CLIENT_IP'])
14      || isset($_SERVER['HTTP_X_FORWARDED_FOR']))
15      || !(in_array(@$_SERVER['REMOTE_ADDR'], ['127.0.0.1', '::1']) || php_sapi_name() === 'cli-server')
16  ) {
17      header('HTTP/1.0 403 Forbidden');
```

Introduced by `agence2dav`, 17 days ago

Time to fix: 5 minutes

Why is this an issue?

Security: Validated Sanitized Input

Related code pattern: [Security: Validated Sanitized Input](#) by `PHP_CodeSniffer`

Il s'agit ici d'une faille anodine, mais qui fait partie des petites choses qui montrent assez typiquement une façon d'écrire le code qui est révolue. L'usage des variables superglobales a toujours été déconseillée, mais les développeurs l'utilisaient parce que c'était pratique.

Ces erreurs semblent peu significatives, mais leur addition et leur exploitation peut conduire à se trouver face à des problèmes qui obligent, de toutes manières, à les résoudre.

### 2.1.7 Audit de performance

Les benchmarks (mesures comparatives avec des points de référence) sont fait en utilisant le *Profiler* de Symfony (voir <https://symfony.com/doc/current/profiler.html>), qui fournit un détail des temps de calcul pour chaque partie du logiciel, de la lecture du code à son rendu final en passant par les diverses étapes de son exécution.

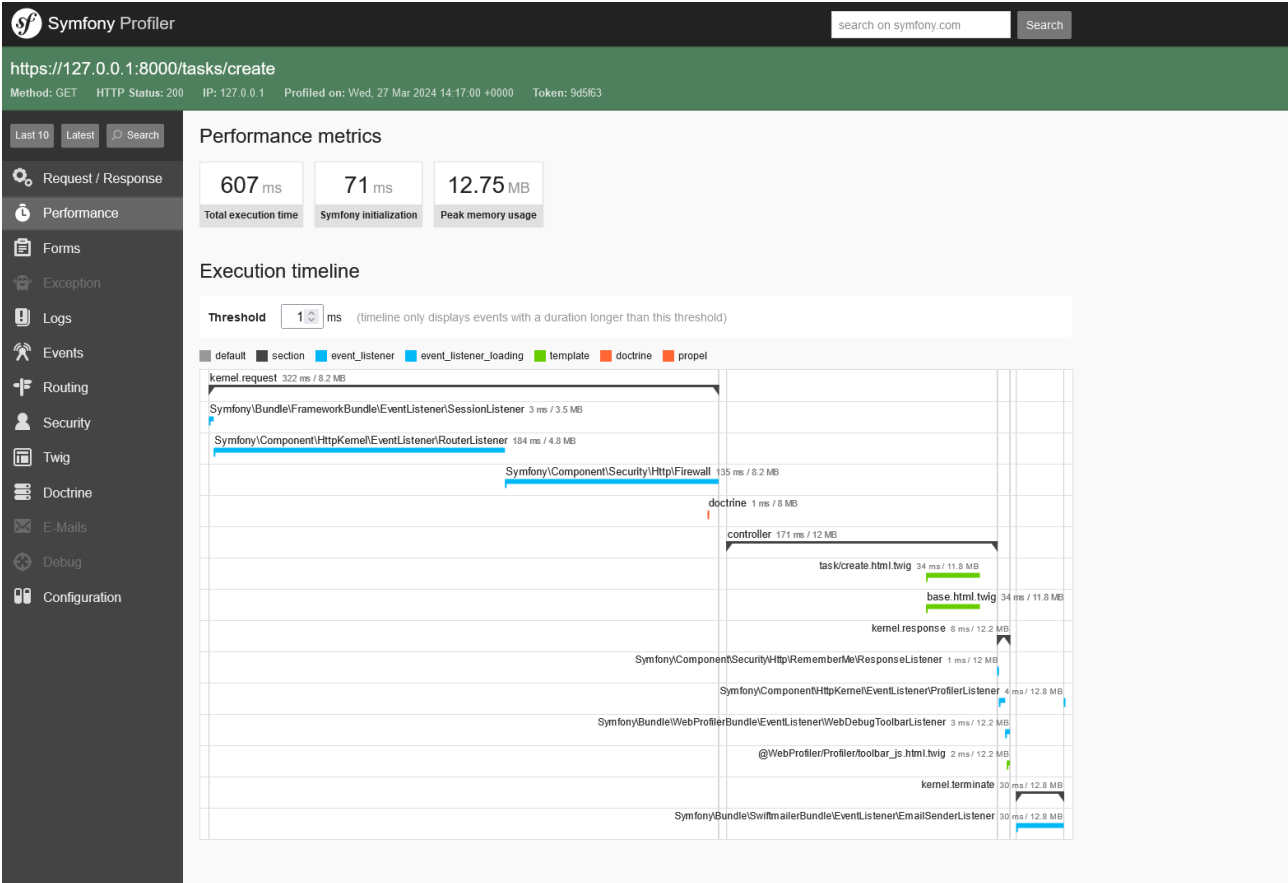


Figure 4: Exemple de résultat du Profiler

Les temps d’exécution ont été faits sur l’essentiel des outes d’accès aux fonctionnalités :

#### 2.1.7.1 /login (post)

total execution (ms)	316
symfony initialization (ms)	73
peak memory usage (Mb)	9,25

#### 2.1.7.2 /tasks (get)

total execution (ms)	318
symfony initialization	72



(ms)	
peak memory usage	
(Mb)	9,5

### **2.1.7.3      /task/id/toggle**

total execution (ms)	308
symfony initialization	
(ms)	70
peak memory usage	
(Mb)	9,5

### **2.1.7.4      /task/create (post)**

total execution (ms)	607
symfony initialization	
(ms)	71
peak memory usage	
(Mb)	12

### **2.1.7.5      /task{id}/edit (post)**

total execution (ms)	318
symfony initialization	
(ms)	72
peak memory usage	
(Mb)	9,5

### **2.1.7.6      /task{id}/delete (get)**

total execution (ms)	314
symfony initialization	
(ms)	67
peak memory usage	
(Mb)	9,5

### **2.1.7.7      Résumé**

En moyenne le logiciel requiert **9,5 Mo** de Ram pour chaque appel et nécessite 0,3 seconde pour son exécution, sur notre environnement de dev.

## 2.2 Plan de remise en route

La résurrection du logiciel peut se faire de deux manières, qui dépendent de sa complexité, en cherchant à optimiser le temps que cela va prendre. Soit on l'upgrade, soit on le refait.

En général on cherche toujours à retarder autant que possible une refonte, afin de collecter l'ensemble des transformations qui vont devoir s'opérer et de s'assurer qu'elle durera plus longtemps.

### 2.2.1 Option 1 : upgrader les versions

Il est approprié d'upgrader pas à pas un logiciel complexe afin de maîtriser chacune de ses composantes, et de marquer des étapes de repos et de stabilisation entre chaque nouvelle mise à niveau.

### 2.2.2 Option 2 : Refonte (From Scratch)

Une refonte est une opération majeure dans la vie d'un site et elle finit toujours par devenir inévitable. C'est l'occasion de réviser ce qui n'a pas été bien pensé dès le début et auquel il aura fallu s'adapter de façon grossière, du supprimer des fonctionnalités inutiles et de mettre l'accent sur celles qui se sont avérées les plus utiles. C'est aussi l'occasion d'avoir de nouvelles idées et une base de travail fraîche sur laquelle l'énergie de développement est renouvelée.

### 2.2.3 Motivation pour la solution adoptée

L'expression « From Scratch » est appropriée étant donné le mal qu'il a fallu se donner pour reproduire les conditions initiales de son fonctionnement. Sans cela, la page qui s'affiche après avoir, seulement installé (pas mis à jour) les composants du /Vendor ressemble à ceci :

```
Warning: require(C:\Users\dav\Documents\srv\git8\app\..\vendor\autoload.php): Failed to open stream: No such file or directory in C:\Users\dav\Documents\srv\git8\app\autoload.php on line 7

Fatal error: Uncaught Error: Failed opening required 'C:\Users\dav\Documents\srv\git8\app\..\vendor\autoload.php' (include_path='.;C:\laragon\etc\php\pear') in C:\Users\dav\Documents\srv\git8\app\autoload.php:7
Stack trace: #0 C:\Users\dav\Documents\srv\git8\app\web\app_dev.php(22): require() #1 {main} thrown in C:\Users\dav\Documents\srv\git8\app\autoload.php on line 7
```

C'est à dire qu'il n'y a même pas d'autoload.

Rien que l'installation des composants aura donné lieu à une longue suite d'erreurs de fonctions dépréciées.

```
Deprecation Notice: trim(): Passing null to parameter #1 ($string) of type string is deprecated in  
C:\Users\dav\Documents\src\oc8\vendor\symfony\symfony\src\Symfony\Component\Yaml\Inline.php:81  
Deprecation Notice: trim(): Passing null to parameter #1 ($string) of type string is deprecated in  
C:\Users\dav\Documents\src\oc8\vendor\symfony\symfony\src\Symfony\Component\Yaml\Inline.php:81  
Deprecation Notice: trim(): Passing null to parameter #1 ($string) of type string is deprecated in  
C:\Users\dav\Documents\src\oc8\vendor\symfony\symfony\src\Symfony\Component\Yaml\Inline.php:81  
Deprecation Notice: trim(): Passing null to parameter #1 ($string) of type string is deprecated in  
C:\Users\dav\Documents\src\oc8\vendor\symfony\symfony\src\Symfony\Component\Yaml\Inline.php:81  
Deprecation Notice: trim(): Passing null to parameter #1 ($string) of type string is deprecated in  
C:\Users\dav\Documents\src\oc8\vendor\symfony\symfony\src\Symfony\Component\Yaml\Inline.php:81  
Deprecation Notice: trim(): Passing null to parameter #1 ($string) of type string is deprecated in  
C:\Users\dav\Documents\src\oc8\vendor\symfony\symfony\src\Symfony\Component\Yaml\Inline.php:81  
Deprecation Notice: trim(): Passing null to parameter #1 ($string) of type string is deprecated in  
C:\Users\dav\Documents\src\oc8\vendor\symfony\symfony\src\Symfony\Component\Yaml\Inline.php:81  
Deprecation Notice: trim(): Passing null to parameter #1 ($string) of type string is deprecated in  
C:\Users\dav\Documents\src\oc8\vendor\symfony\symfony\src\Symfony\Component\Yaml\Inline.php:81
```

Dans un état vaguement fonctionnel, on obtient ceci :

Whoops, looks like something went wrong.

1/1

**FatalThrowableError** in ErrorHandler.php line 364:

Type error: Too few arguments to function

Symfony\Component\Debug\ErrorHandler::handleError(), 4 passed in C:

\Users\dav\Documents\src\oc8\app\AppKernel.php on line 6 and at least 5 expected

1. in ErrorHandler.php line 364
2. at ErrorHandler->handleError('8192', 'AppKernel implements the Serializable interface, which is deprecated. Implement \_\_serialize() and \_\_unserialize() instead (or in addition, if support for old PHP versions is necessary)', 'C:\Users\dav\Documents\src\oc8\app\AppKernel.php', '6') in AppKernel.php line 6
3. at require\_once('C:\Users\dav\Documents\src\oc8\app\AppKernel.php') in DebugClassLoader.php line 142
4. at DebugClassLoader->loadClass('AppKernel') in app\_dev.php line 25

C'est à dire que la version actuelle de Php ne prend simplement pas en charge le code du projet.

Finalement, après avoir dû Downgrader en deux étapes l'environnement (passer à Php 7.4 pour installer Symfony 3 puis installer une version fonctionnelle de Php-5.5.13) on a pu obtenir le site tel qu'il était laissé, afin d'y faire nos tests.

Étant donné que nous n'avions que deux semaines, et la faible complexité du logiciel, qui ne contient que deux principaux éléments (les Users et les Tasks), il était plus efficace de partir directement sur une version neuve de Symfony 7 sous Php-8.3.4 (actuel) et de s'adapter à cela (voir documentation technique).

### 3 Nouvelle version V2

Un projet Symfony fraîchement installé dispose d'office de l'ensemble des dispositifs de sécurité et de gestion des bases de données dont nous avons besoin.

## 3.1 Le projet modifié

Une fois la logiciel opérationnel il aura fallu appliquer un certain nombre de correctifs et ajouter des fonctionnalités (Voir document technique).

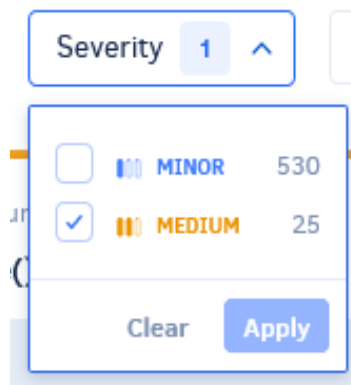
Une fois ceci fait, on a pu récolter le fruit de notre labeur en mesurant les indices de qualité et de performance suivants :

### 3.1.1 Audit de qualité

L'audit de qualité de Codacy atteste avoir passé les épreuves de qualité et de sécurité.

(Image manquante)

#### 3.1.1.1 Sévérité des erreurs



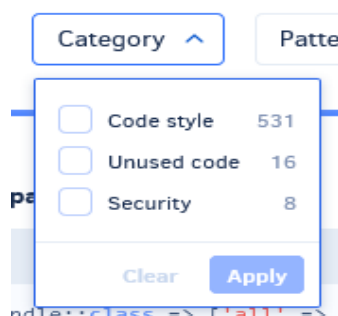
The image shows a user interface for filtering error severity. At the top, there is a 'Severity' dropdown menu currently set to '1'. Below it, a list of options is displayed: 'MINOR' with a count of 530 and 'MEDIUM' with a count of 25. The 'MEDIUM' option is selected, indicated by a checkmark in a box. At the bottom of the list, there are two buttons: 'Clear' and 'Apply'.

Severity	Count
MINOR	530
MEDIUM	25

Figure 5: Sévérité des erreurs

Le rapport fait état de **0** erreur critiques :

minor	530
medium	25
critical	0



Category	Count
Code style	531
Unused code	16
Security	8

Figure 6: Catégories d'erreurs

Au total, **6** erreurs portent sur la sécurité. Ce sont des erreurs de sévérité moyenne, qui portent sur des composants de Symfony lui-même et non sur notre code source :

# Projet ToDo & Co – Audit de qualité et de performance

Go back

9

🏠

🔍

📄

🔔

📁

🛡️

📊

🔧

🔍

🕒

agency2dav

oc8

🔔

Community

Docs

🔔

Enhance code security with Semgrep!

Discover and fix security issues before you merge. Activate Semgrep to find and eliminate vulnerabilities in your code. [Apply Semgrep to Coding Standards.](#)

✕

Issues

v2

About this page

🔔 Current 555

🕒 Ignored 0

Filter by

Language

Severity 1

Category 1

Pattern

Author

Clear all

MEDIUM

Security

Insecure Modules Libraries

...

✕

The use of function `dirname()` is discouraged

config/preload.php

4

`require dirname(__DIR__).'../var/cache/prod/App_KernelProdContainer-preload.php';`

MEDIUM

Security

...

✕

"require\_once" statement detected. File manipulations are discouraged. Concatenating is forbidden.

public/index.php

5

`require_once dirname(__DIR__).'../vendor/autoload_runtime.php';`

MEDIUM

Security

Insecure Modules Libraries

...

✕

The use of function `dirname()` is discouraged

public/index.php

5

`require_once dirname(__DIR__).'../vendor/autoload_runtime.php';`

MEDIUM

Security

...

✕

"require" statement detected. File manipulations are discouraged. Concatenating is forbidden.

tests/bootstrap.php

5

`require dirname(__DIR__).'../vendor/autoload.php';`

MEDIUM

Security

Insecure Modules Libraries

...

✕

The use of function `dirname()` is discouraged

tests/bootstrap.php

10

`(new Dotenv())->bootEnv(dirname(__DIR__).'../.env');`

MEDIUM

Security

Insecure Modules Libraries

...

✕

The use of function `dirname()` is discouraged

tests/bootstrap.php

8

`require dirname(__DIR__).'../config/bootstrap.php';`

MEDIUM

Security

...

✕

"require" statement detected. File manipulations are discouraged. Concatenating is forbidden.

tests/bootstrap.php

8

`require dirname(__DIR__).'../config/bootstrap.php';`

MEDIUM

Security

Insecure Modules Libraries

...

✕

The use of function `dirname()` is discouraged

tests/bootstrap.php

5

`require dirname(__DIR__).'../vendor/autoload.php';`

### 3.1.2 Audit de performance

Les temps d'exécution ont été faits sur l'essentiel des routes d'accès aux fonctionnalités :

#### 3.1.2.1 */login (post)*

total execution (ms)	341
symfony initialization (ms)	146
peak memory usage (Mb)	26

#### 3.1.2.2 */tasks (get)*

total execution (ms)	371
symfony initialization (ms)	143
peak memory usage (Mb)	30

#### 3.1.2.3 */task/id/toggle*

total execution (ms)	369
symfony initialization (ms)	138
peak memory usage (Mb)	26

#### 3.1.2.4 */task/create (post)*

total execution (ms)	334
symfony initialization (ms)	139
peak memory usage (Mb)	26

#### 3.1.2.5 */task{id}/edit (post)*

total execution (ms)	385
symfony initialization (ms)	151
peak memory usage (Mb)	30

### 3.1.2.6 /task{id}/delete (get)

total execution (ms)	334
symfony initialization (ms)	139
peak memory usage (Mb)	26

### 3.1.2.7 *Résumé*

En moyenne le logiciel requiert **28 Mo** de Ram pour chaque appel et nécessite 0,3 seconde pour son exécution, sur notre environnement de dev.

## 4 Synthèse

### 4.1 Mesures comparative de la qualité

Le niveau de qualité est indéniablement en faveur de la V2.

Le nombre d'erreurs mineures, qui a plus que doublé, est imputable au fait d'avoir produit un code plus explicite, plus déployé et plus long, d'avoir ajouté des fonctionnalités, et d'avoir encore doublé cette quantité de code par des tests unitaires.

		Version 1	Version 2
<b>severity errors</b>	minor	245	530
	medium	21	25
	critical	18	0
<b>category errors</b>	code style	251	531
	unused code	3	16
	security	30	8



Le résultat est très largement en faveur de la V2.

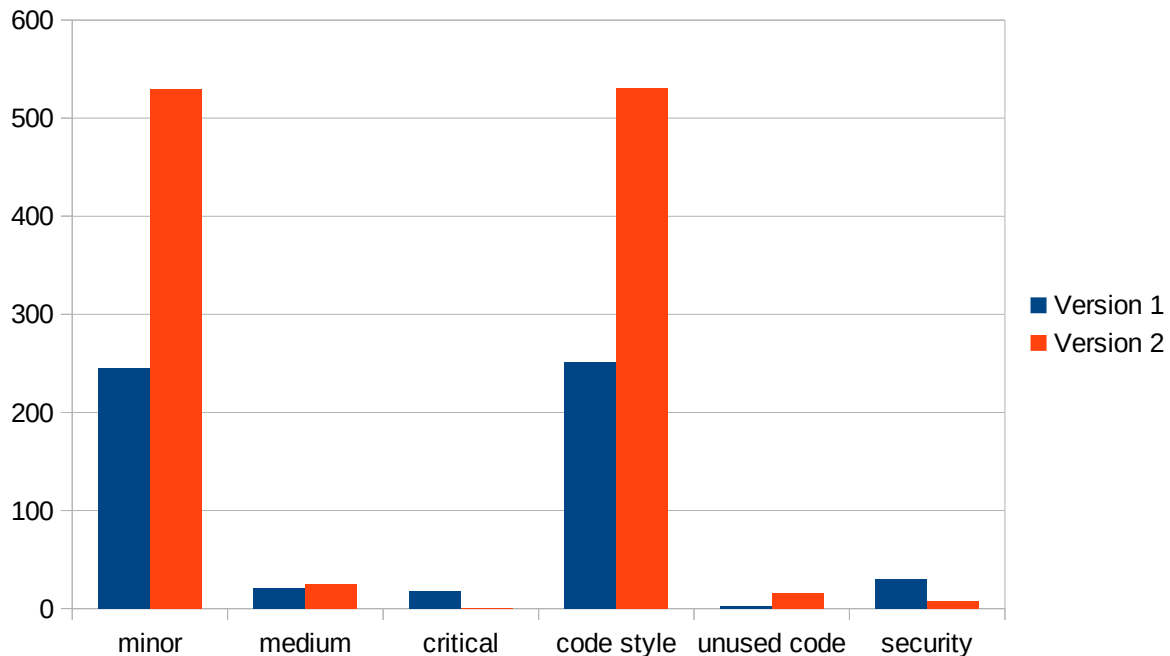


Figure 7: Schéma comparatif des mesures de qualité

## 4.2 Mesures comparative des performances

Comme déjà précisé, le progrès informatique montre que le gain de performance et généralement compensé par le gain de complexité, de sorte que les logiciels ne semblent pas aller « plus vite », voire même risquent de perdre un peu en vitesse par rapport à leurs ancêtres.

Il est notable que la quantité de Ram exigée pour chaque exécution est multipliée par **trois**, ce qui est un défaut majeur, qui pourrait être rédhibitoire, ou bien avoir un impact significatif sur le coût du serveur. En effet s'il faut supporter cent appels simultanés, cela représente 3 Go de RAM au strict minimum, qui seront constamment alimentés en énergie. Mais heureusement il ne s'agit pas d'un site destiné à avoir un fort trafic.

total execution (ms)	363	355
symfony initialization (ms)	70	142
peak memory usage (Mb)	9,825	27,3

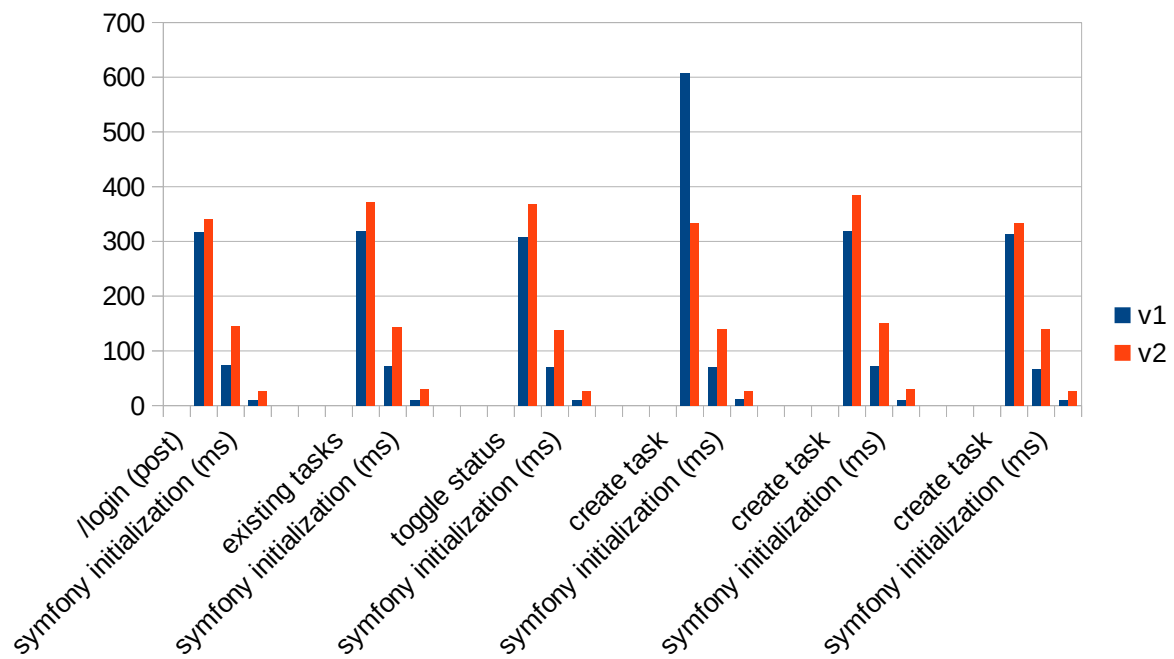


Figure 8: Schéma comparatif des mesures de performance

## 5 Conclusion

La refonte du logiciel était la décision la plus économique en terme de temps et a produit les résultats escomptés en terme de qualité et de performance.

