

Rapport —

ANALYSE DE LA QUALITE DE CODE ET DE LA PERFORMANCE





Table des matières

Contexte	3
Outils Utilisés.....	3
État des lieux	4
Framework Symfony.....	4
PHP.....	5
Base de données MySQL	5
Les besoins.....	5
Corrections d'anomalies	5
Implémentation de nouvelles fonctionnalités.....	6
Analyse du code initial	7
1- Code Climate	7
2- Codacy	7
Analyse des performances	9
Solutions mises en place	10
1- Mise à jour des versions non maintenues	10
2- Anomalies de fonctionnements	10
a- Listes des anomalies (en plus des anomalies rapportées).....	10
b- Corrections des anomalies	10
c- Corrections des anomalies présente dans l'application.....	11
3- Ajout de nouvelles fonctionnalités	12
a- Autorisation.....	12
b- Implémentation des Tests Automatisés.....	12
4- Analyse des performances	14
5- Propositions d'améliorations.....	16



Contexte

ToDo&Co est une startup dont le cœur de métier est une application permettant de gérer ses tâches quotidiennes (ToDoList).

Pour montrer à de potentiels investisseurs le potentiel de l'application, l'entreprise a développé l'application en se basant sur l'essentiel (on parle de Minimum Viable Product ou MVP).

Le choix du Framework PHP Symfony est déjà défini lors du démarrage du projet.

Le projet étant maintenant lancé suite à la levée de fonds, le but actuel est d'améliorer la qualité de l'application.

La qualité englobe :

- Qualité de code
- Qualité perçue (performance)
- Qualité niveau maintenabilité.

Les moyens à mettre en œuvre :

- L'implémentation de nouvelles fonctionnalités ;
- La correction de quelques anomalies ;
- L'implémentation de tests automatisés.

Outils Utilisés

Mesure de performance => Profiler Symfony

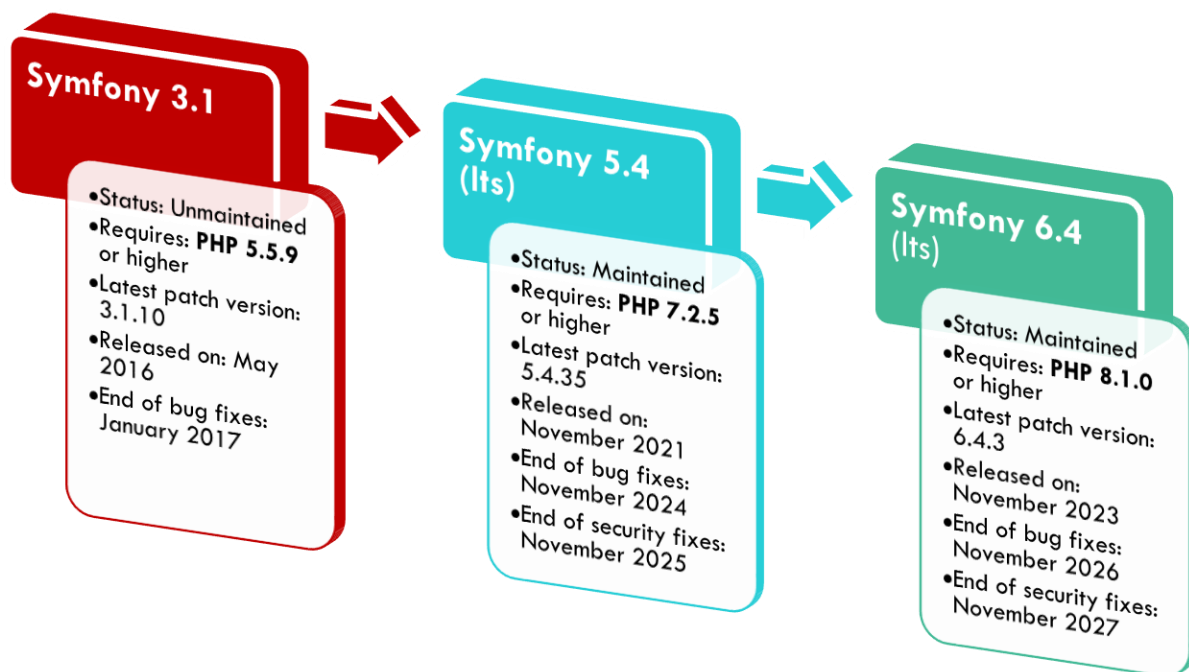
Qualité du code

1. PhpStan => Outil d'analyse PHP pour détecter les erreurs
2. Php-cs-fixer => Outil de correction de la syntaxe PHP
3. Codacy => Outil de revue de code automatisé qui aide les développeurs à gagner du temps dans les revues de code et à s'attaquer efficacement à la dette technique.
4. CodeClimate => Outil d'analyse du code sur la base de diverses mesures, notamment la complexité du code, la duplication, la couverture des tests, les vulnérabilités en matière de sécurité et le respect des meilleures pratiques.



État des lieux

Framework Symfony



L'application a été développée avec la version 3.1 du framework Symfony. Cette version n'est plus maintenue depuis janvier 2017 pour les bugs et depuis juillet 2017 pour la sécurité. Les composants Symfony et les bibliothèques tierces ne sont également plus à jour voir sont dépréciés.

Il est donc nécessaire d'effectuer la mise à jour du framework vers une version LTS (Long Term Support) .

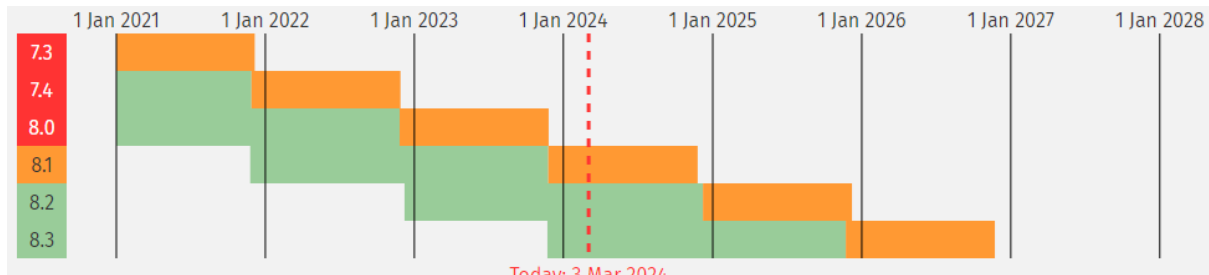
Deux possibilités s'offrent à nous :

- Soit la version 5.4 qui était l'ancienne LTS
- Soit la version 6.4 la dernière LTS en date.



PHP

Concernant la version de PHP, le projet utilise la version 5.59 qui n'est pas maintenue. L'application s'expose donc à des risques de sécurité. Le projet doit évoluer vers une version php supportée (celle-ci sera choix en fonction de la version du Framework Symfony)



Base de données MySQL

La base de données utilisée par le projet est la version Mysql 5.7 , une version qui est en fin de vie depuis octobre 2023.

Release	Release date	End of life	
MySQL 8.0	April 19, 2018	April, 2026	
MySQL 5.7	October 21, 2015	October 21, 2023	

Afin de rester dans l'optique de version supportée, le passage à la version 8.0 est indispensable

Les besoins

Corrections d'anomalies

1. Une tâche doit être attachée à un utilisateur

Actuellement, lorsqu'une tâche est créée, elle n'est pas rattachée à un utilisateur. Il vous est demandé d'apporter les corrections nécessaires afin qu'automatiquement, à la sauvegarde de la tâche, l'utilisateur authentifié soit rattaché à la tâche nouvellement créée.

2. Lors de la modification de la tâche, l'auteur ne peut pas être modifié.
3. Pour les tâches déjà créées, il faut qu'elles soient rattachées à un utilisateur "anonyme".



4. Choisir un rôle pour un utilisateur

Lors de la création d'un utilisateur, il doit être possible de choisir un rôle pour celui-ci. Les rôles listés sont les suivants :

- Rôle utilisateur (ROLE_USER) ;
- Rôle administrateur (ROLE_ADMIN).

5. Lors de la modification d'un utilisateur, il est également possible de changer le rôle d'un utilisateur.

Implémentation de nouvelles fonctionnalités

Autorisation

1. Seuls les utilisateurs ayant le rôle administrateur (ROLE_ADMIN) doivent pouvoir accéder aux pages de gestion des utilisateurs.
2. Les tâches ne peuvent être supprimées que par les utilisateurs ayant créé les tâches en question.
3. Les tâches rattachées à l'utilisateur "anonyme" peuvent être supprimées uniquement par les utilisateurs ayant le rôle administrateur (ROLE_ADMIN).

Implémentation de tests automatisés

L'implémentation de tests automatisés (tests unitaires et fonctionnels) est nécessaire pour assurer que le fonctionnement de l'application est bien en adéquation avec les demandes.

Ces tests seront implémentés grâce à PHPUnit.

Vous prévoyez des données de tests afin de pouvoir prouver le fonctionnement dans les cas explicités dans ce document.

Pour le rapport de couverture de code au terme du projet, un taux de couverture supérieur à 70 % est attendu.

Analyse du code initial

1- Code Climate

Breakdown

56 FILES



TEST COVERAGE

Codebase summary

MAINTAINABILITY



TEST COVERAGE



Repository stats

CODE SMELLS

26

DUPLICATION

36

OTHER ISSUES

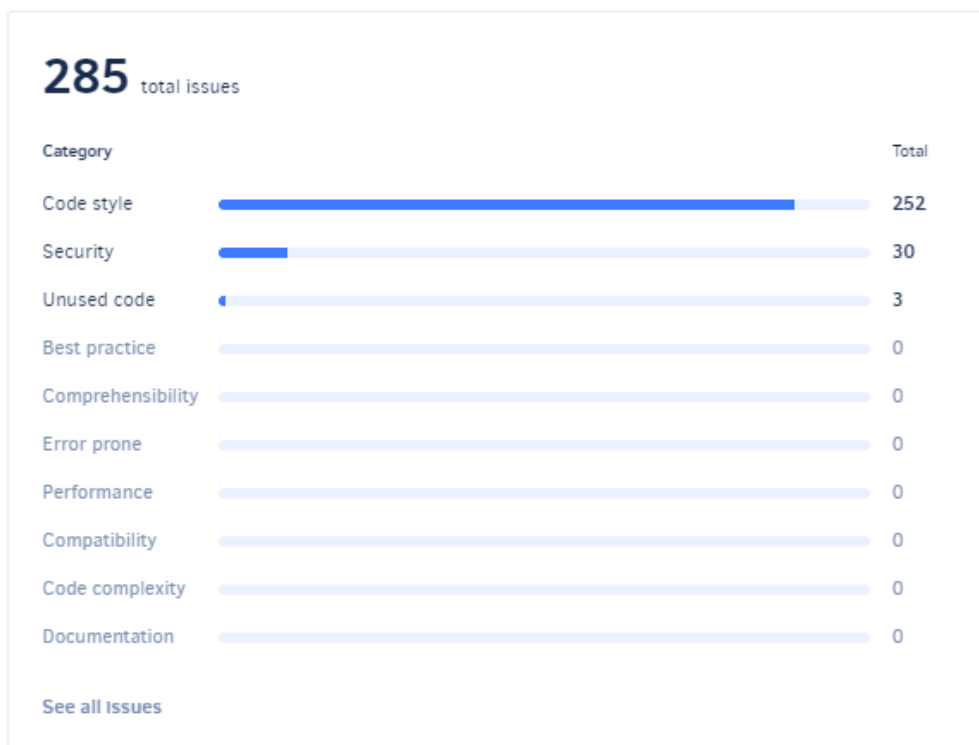
0

Lors de l'analyse de CodeClimate, nous pouvons noter que le code contient beaucoup de duplications et de codes pouvant être améliorés.

2- Codacy

Ci-Dessous nous avons le nombre de corrections à apporter au code.

L'étendue de ces corrections va de la simple mauvaise indentation à des problèmes de Sécurité (utilisation de super-globale).





Exemple d'issues avec une sévérité critique :

CRITICAL	Security	Insecure Modules Libraries
Insecure dependency swiftmailer/swiftmailer@v5.4.3 (CVE-2016-10074: The mail transport (aka Swift_Transport_MailTransport) in Swift Mailer ...) (update to 5.0.0, 5.4.5)		
composer.lock		
1255	{	

CRITICAL	Security	
Use of exit language construct is discouraged.		
web/app_dev.php		
18	exit('You are not allowed to access this file. Check '.basename(__FILE__).' for more information.');	

CRITICAL	Security	
Direct use of \$_SERVER Superglobal detected.		
web/app_dev.php		
15	!(in_array(\$_SERVER['REMOTE_ADDR'], ['127.0.0.1', '::1'])) php_sapi_name() === 'cli-server'	

CRITICAL	Security	
Direct use of \$_SERVER Superglobal detected.		
web/app_dev.php		
14	isset(\$_SERVER['HTTP_X_FORWARDED_FOR'])	

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		
15	!(in_array(\$_SERVER['REMOTE_ADDR'], ['127.0.0.1', '::1'])) php_sapi_name() === 'cli-server'	

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		
18	exit('You are not allowed to access this file. Check '.basename(__FILE__).' for more information.');	

CRITICAL	Security	
Direct use of \$_SERVER Superglobal detected.		
web/app_dev.php		
13	if (isset(\$_SERVER['HTTP_CLIENT_IP'])	

CRITICAL	Security	
Use of echo language construct is discouraged.		
web/config.php		
376	<p class="help"><?php echo \$problem->getHelpHtml() ?></p>	

Beaucoup d'améliorations de sécurité seront effectuées lors de l'upgrade des versions.



Analyse des performances

Le profiler de Symfony a été utilisé pour l'analyse de performance de l'application.

Pour mesurer la performance, 3 catégories de données ont été collectées :

- Performance metrics : temps total d'exécution, temps d'initialisation de Symfony, quantité de mémoire allouée
- Template metrics : temps pour l'affichage de la vue, nombre de bloc Twig
- Doctrine metrics : nombre de requêtes SQL exécutées, temps d'exécution

Ci-dessous un aperçu du fichier Excel regroupant les données des différentes routes (GET et POST) :

Version	position	URI	Route	Methode	Performance Metrics			Template Metrics		Query Metrics	
					Total Execution Time	Symfony Initializati	Peak Memory Usa	Render Time (m	Template	Queries Time (m	Query Cal
3.1	1	/	login	GET	246	106	4	7	4	0	0
3.1	2	/users/create	user_create	GET	101	6	6	50	2	0	0
3.1	3	/users/create	user_create	POST	699	11	4	0	0	13	2
3.1	4	/users	users	GET	32	7	2	1	2	1	1
3.1	5	/login	login	GET	21	8	2	0	0	0	0
3.1	6	/login	login_check	POST	474	5	2	0	0	1	1
3.1	7	/	homepage	GET	31	4	2	0	0	1	1
3.1	8	/tasks	tasks	GET	41	8	2	1	2	2	2

On peut remarquer un allongement du temps de traitement lors de la validation de formulaires (méthodes POST), certainement dû au système de hachage de mot de passe et à la validation des différentes données soumises.

Il faut noter qu'aucune pagination n'est présente dans les visualisations des listes d'utilisateurs et de tâches, donc le temps de traitement augmentera quand le nombre de données à afficher sera plus important.



Solutions mises en place

1- Mise à jour des versions non maintenues

Comme indiqué précédemment, l'application a évolué vers la dernière version LTS de Symfony : la version 6.4.

Les dépendances sont alors également mises à jour.

Les versions dépréciées des dépendances ont été remplacées (ex : swiftmailer)

La version de PHP a également été mise à jour vers la version 8.1.

La version de MySQL a également été mise à jour vers la version 8.0.

Cf Annexe – Upgrade to lts version.

Concernant la partie Front-End, la version de Bootstrap a été mise à jour (3.3.7 vers 5.3.3).

Cette version de Bootstrap n'utilise plus JQuery pour son fonctionnement ce qui entraîne donc un gain de temps pour l'affichage des pages.

Tout fonctionne mais un ajustement devra être effectué pour avoir un visuel plus actuel.

2- Anomalies de fonctionnements

a- Listes des anomalies (en plus des anomalies rapportées)

Après examen initial du projet, d'autres anomalies ont pu être identifiées :

- Lien vers la page d'accueil inexistant
- Lien vers la page liste des utilisateurs inexistant
- Page liste des tâches terminées inexistante
- Page liste des tâches à faire comprend toutes les tâches
- Aucun Message flash pour la modification d'une tâche en 'marquée non terminée'
- Manque la gestion du pseudo unique
- Absence validation mot de passe
- Absence de token CSRF dans les formulaires.
- Absence de page d'erreurs (403-401)
- Absence de JQuery pour l'utilisation de Bootstrap 3.3.7

b- Corrections des anomalies

- a- Une tâche doit être attachée à un utilisateur (issues #18)
 - ❖ Ajout jointure entre les utilisateurs et les taches (Entity)
 - ❖ Ajout de l'utilisateur connecté à la création d'une tâche (Controller)



- b- Lors de la modification de la tâche, l'auteur ne peut pas être modifié. (issues #19)
 - ❖ Aucune modification effectuée
- c- Pour les tâches déjà créées, il faut qu'elles soient rattachées à un utilisateur "anonyme".(issues #20)
 - ❖ Ajout de la possibilité d'un utilisateur 'null' => signalé 'anonyme' dans l'application
 - ❖ Ajout visualisation dans les templates
- d- Choisir un rôle pour un utilisateur lors de la création et de la modification (issues #21)
 - ❖ Ajout de l'information dans la base de données (Entity)
 - ❖ Ajout de la sélection du rôle dans le formulaire de création d'utilisateur
 - ❖ Modification du Controller (méthodes)

c- Corrections des anomalies présente dans l'application

1. Lien vers la page d'accueil inexistant (issues #27)
 - ❖ Ajout d'un lien dans le template de base
2. Lien vers la page liste des utilisateurs inexistant (issues #29)
 - ❖ Ajout d'un lien dans le template de la page d'accueil
3. Page liste des tâches terminées inexistante (issues #31)
 - ❖ Ajout du lien de la nouvelle page
 - ❖ Création de la méthode (Controller)
 - ❖ Modification des liens de retour (supprimer , marqué fait)
4. Page liste des tâches à faire comprend toutes les tâches (issues #29)
 - ❖ Filtrer les pages dans le Controller
5. Aucun Message flash pour la modification d'une tâche en 'marquée non terminée' (issues #31)
 - ❖ Ajout du message flash de validation
6. Manque la gestion du pseudo unique (issues #33)
 - ❖ Ajout unique
7. Absence validation mot de passe (issues #35)
 - ❖ Ajout d'un regex
8. Absence de page d'erreurs (403-401) (issues #46)
 - ❖ Ajout de Listener et Handler.
9. Absence de JQuery pour l'utilisation de Bootstrap 3.3.7
 - ❖ Disparition de l'utilisation de JQuery avec la version de Bootstrap 5.3
10. Absence de vérification d'autorisation d'accès aux pages (Issue #49)
 - ❖ Ajout d'une couche de sécurité

Une amélioration du code suivant les erreurs de PhpStan et un nettoyage des fichiers inutilisés suivant les différentes montées de versions s'imposent.



3- Ajout de nouvelles fonctionnalités

a- Autorisation

Nous avons effectué la mise en place des fonctionnalités suivantes :

1. Seuls les utilisateurs ayant le rôle administrateur (ROLE_ADMIN) doivent pouvoir accéder aux pages de gestion des utilisateurs.
2. Les tâches ne peuvent être supprimées que par les utilisateurs ayant créé les tâches en question.
3. Les tâches rattachées à l'utilisateur "anonyme" peuvent être supprimées uniquement par les utilisateurs ayant le rôle administrateur (ROLE_ADMIN).

L'utilisation d'un Voter (pour la gestion des accès) pour valider la possibilité de suppression d'une tâche et l'ajout d'une permission (IsGranted) pour l'accès à la page des utilisateurs ont été nécessaires pour effectuer ces modifications.












Une modification des templates a été faite pour avoir un visuel plus lisible pour tout le monde.

- ❖ Accès à la page utilisateurs de la navbar(seulement pour ROLE_ADMIN)
- ❖ Identification des tâches sans utilisateur associé

b- Implémentation des Tests Automatisés

Des tests fonctionnels ont été implémentés dans l'application avant d'effectuer les corrections et l'ajout des fonctionnalités

Ci-dessous, le tableau représente la couverture (Coverage) des tests avant les modifications.

	Code Coverage							
	Lines		Functions and Methods		Classes and Traits			
Total		95.05% 96 / 101		87.18% 34 / 39		70.00%		7 / 10
■ Controller		96.61% 57 / 59		83.33% 10 / 12		75.00%		3 / 4
■ Entity		87.50% 21 / 24		86.96% 20 / 23		0.00%		0 / 2
■ Form		100.00% 16 / 16		100.00% 2 / 2		100.00%		2 / 2
■ Repository		100.00% 2 / 2		100.00% 2 / 2		100.00%		2 / 2

Legend

Low: 0% to 35% Medium: 35% to 70% High: 70% to 100%



Après les modifications les tests ont été ajustés pour valider les modifications, ci-dessous les nouveaux tests :

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total	<div><div></div></div>	93.63%	147 / 157	<div><div></div></div>	86.79%	46 / 53	<div><div></div></div>	66.67%	10 / 15
■ Controller	<div><div></div></div>	97.37%	74 / 76	<div><div></div></div>	85.71%	12 / 14	<div><div></div></div>	80.00%	4 / 5
■ Entity	<div><div></div></div>	88.00%	22 / 25	<div><div></div></div>	87.50%	21 / 24	<div><div></div></div>	0.00%	0 / 2
■ EventListener	<div><div></div></div>	62.50%	5 / 8	<div><div></div></div>	66.67%	2 / 3	<div><div></div></div>	0.00%	0 / 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%	2 / 2	<div><div></div></div>	100.00%	2 / 2	<div><div></div></div>	100.00%	2 / 2
■ Security	<div><div></div></div>	90.48%	19 / 21	<div><div></div></div>	87.50%	7 / 8	<div><div></div></div>	66.67%	2 / 3

Nous pouvons voir qu'une grande majorité des méthodes ont un test de vérification.

L'objectif d'un minimum de 70% de couverture est donc atteint avec un taux de couverture de plus de 80%.

Pour atteindre ce taux de couverture, 35 tests comprenant un total de 131 assertions ont été implémentés.

```

PHPUnit 9.6.17 by Sebastian Bergmann and contributors.

Random Seed:  1709935300

Testing
.....                                     35 / 35 (100%)

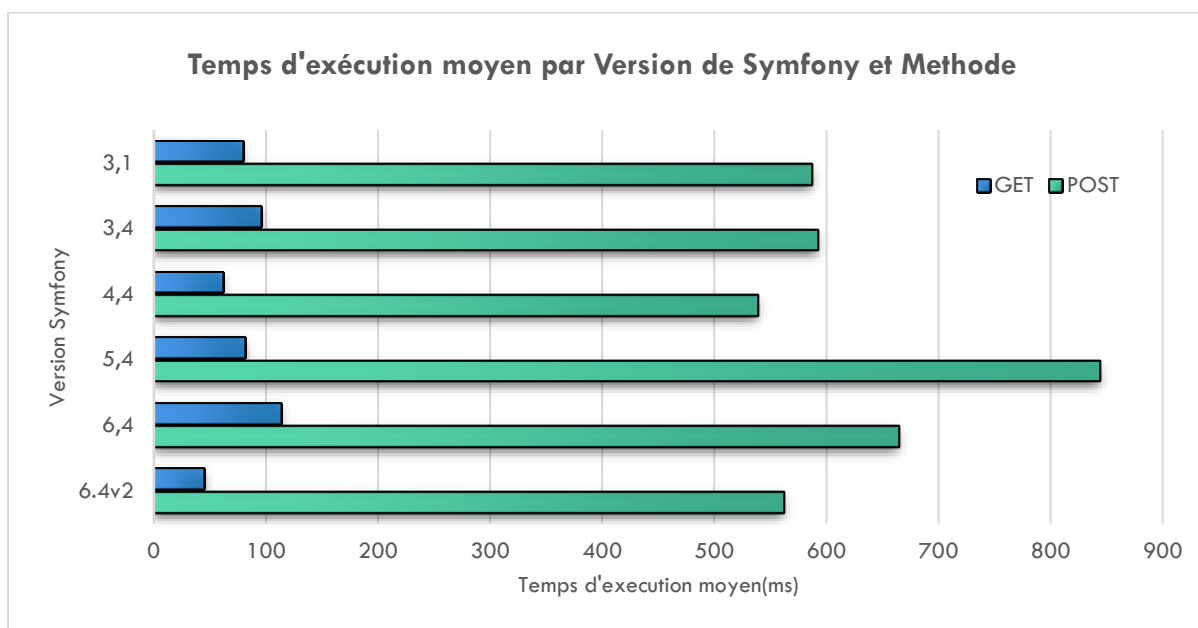
Time: 00:07.422, Memory: 74.50 MB

OK (35 tests, 131 assertions)

```



4- Analyse des performances

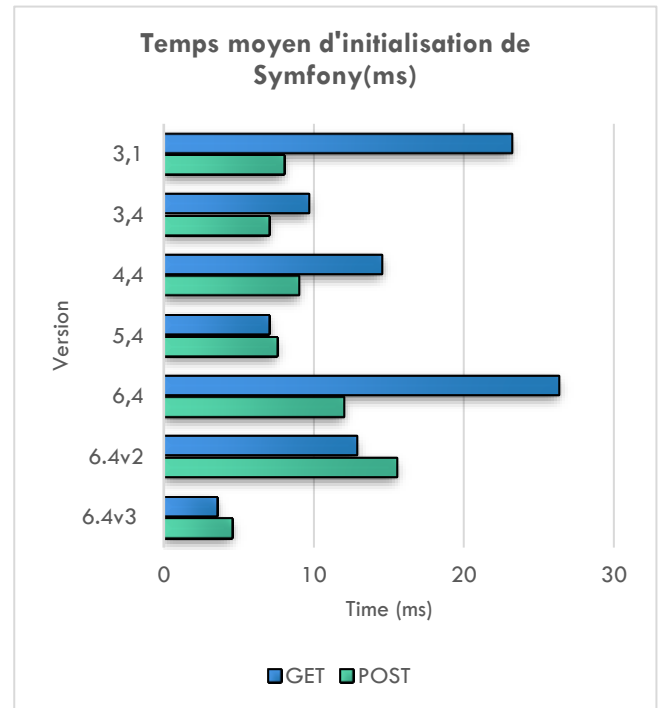
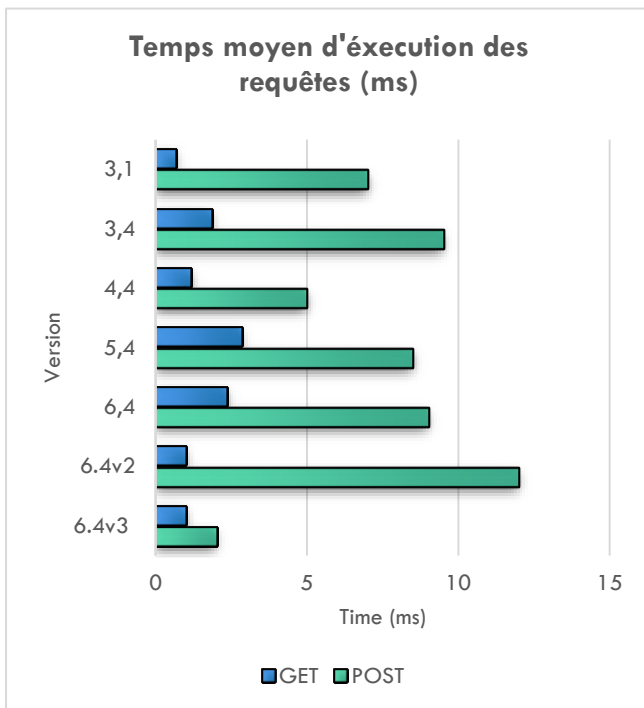


Pour le moment nous ne nous sommes pas attaqués aux gains de performances de notre application.

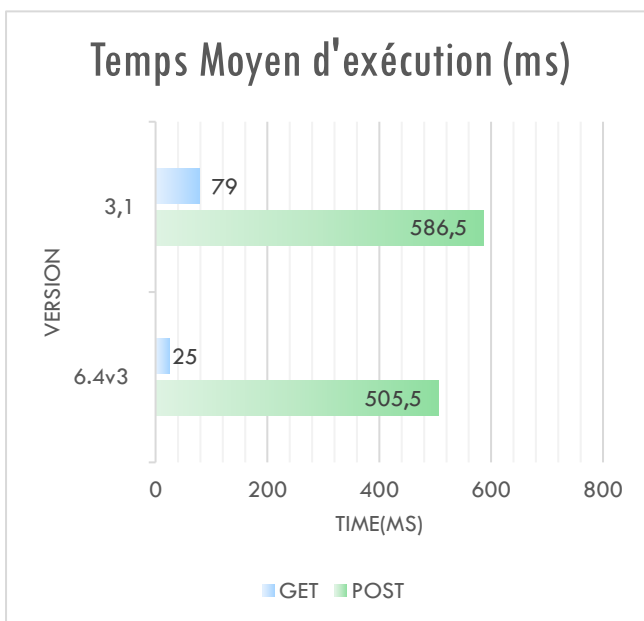
Mais nous pouvons remarquer que la montée de versions de PHP/Symfony n'a pas permis une diminution significative du temps de traitement.

Malheureusement l'outil de WebProfiler de Symfony ne permet pas d'avoir une étude fine mais afin d'améliorer la réactivité de l'application nous avons mis en place un système de cache au niveau des fichiers de configuration du serveur(OPCache) afin de garder en mémoire les actions récurrentes.

- 6.4v2 : version Symfony 6.4 avec fonctionnalités supplémentaires.
- 6.4v3 : Version 6.4v2 avec OPCache



Ces graphiques nous montrent l'amélioration du temps moyen de traitement des requêtes et d'initialisation de Symfony.



Grace à l'Upgrade en version lts de Symfony, PHP et MySQL, et l'ajout d'OPCache, les temps d'exécution se sont sensiblement réduits.

Mais comme indiqué ci-dessus, le Web Profiler de Symfony n'est pas la meilleure solution pour effectuer ce type de comparaison, pour un travail plus fin, l'utilisation de la version payante de blackfire.io sera d'une grande aide.



5- Propositions d'améliorations

Afin de continuer à améliorer l'application, voici quelques propositions d'améliorations :

- Améliorer le visuel de l'application
- Mettre une date d'échéance pour les tâches
- Mettre en place un Dashboard avec les tâches proches de l'échéance, et celles qui l'ont dépassées
- Possibilité de Catégoriser les tâches.
- Mettre en place un rappel par email si une tâche n'est pas terminée avant échéance
- Mettre en place une pagination pour les tâches et les utilisateurs