

TO DO & CO

—

**RAPPORT D'AUDIT DE QUALITE DE CODE ET DE
PERFORMANCE**

1) Constat :

Le projet tel que fournit actuellement repose sur une version dépréciée de Symfony (3.1) et PHP (5.5). Cette situation expose notre projet à de nombreuses vulnérabilités, ainsi que des problèmes de performances.

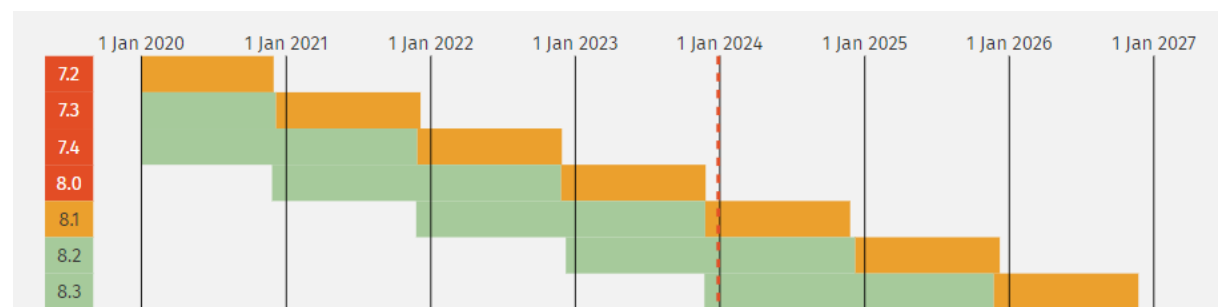
Notre code n'est plus à jour depuis 7 ans et, si nous suivons la roadmap de Symfony, notre version est, à ce jour, plus maintenue :



Au vu de ce graphique, il apparaît opportun de mettre à niveau notre version de Symfony vers celle qui offrira la stabilité la plus importante. A savoir Symfony 6.4 qui offre une version maintenue au-delà de 2026 ce qui assurera la pérennité de notre application.

Cette version 6.4 est ce que l'on appelle la version Long Term Support Release. Elle recevra des mises à jour de sécurité et des corrections de bugs pendant une période plus longue que les versions standard, c'est donc la version la plus stable à ce jour.

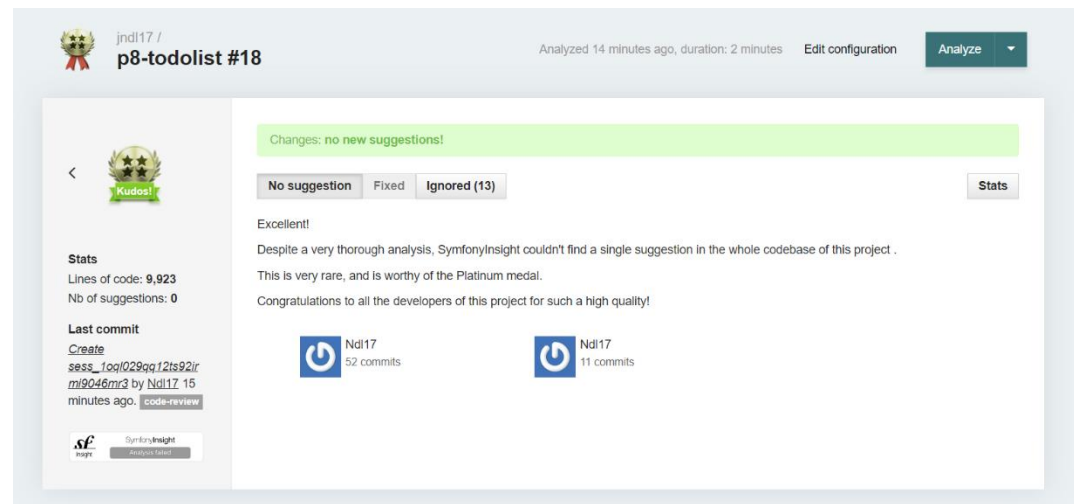
Avec bien entendu une version de PHP la plus à jour possible à savoir 8.3. Ce qui permettra d'améliorer grandement les performances par rapport à notre version actuelle.



C'est pourquoi, afin de partir sur une base saine et éviter au maximum tout relitac de fonctionnalités dépréciées, il est apparu plus judicieux de recommencer un nouveau projet en version 6.4 de Symfony.

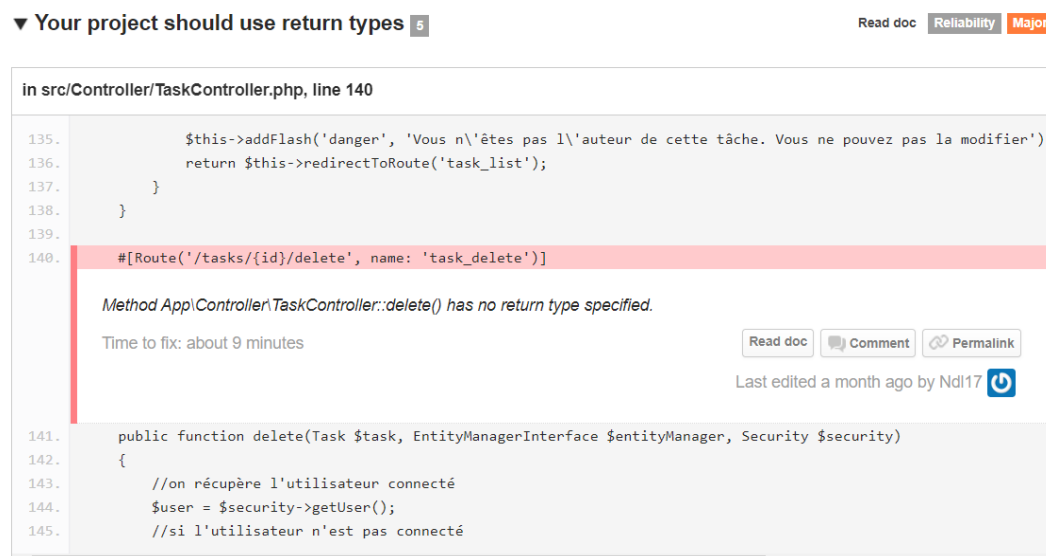
2) Analyse qualité :

Une analyse de la qualité du code a été produite grâce à l'outil « Symfony Insight ». Celle-ci a été réalisée pour finaliser le projet et s'assurer de la qualité générale du code. Cette analyse aura permis de corriger certaines erreurs commises dans le code et de valider la fiabilité de celui-ci.



Notre rapport final d'analyse note notre projet en niveau « Platinum » à savoir le niveau maximal de qualité de la plateforme Symfony Insight.

Ci-dessous, vous trouverez un exemple d'erreur rencontré durant l'analyse de la qualité du code. Par exemple ici, le type de retour attendu pour la fonction delete() du TaskController n'était pas spécifié.



Suite à ce retour d'erreur, le type de retour a pu être rajouté à la fonction. Ce qui permet d'éviter les erreurs à l'exécution en garantissant que la fonction renvoie toujours une valeur du type attendu. Mais aussi d'améliorer la lisibilité du code pour comprendre rapidement ce que la fonction est censée renvoyer.

3) Analyse des performances :

Un audit de performance a été réalisé grâce au profiler de Symfony et un relevé de temps de chargement et de mémoire a été produit sur plusieurs pages de l'application. Les relevés ont été faits sur le projet initial avec la version de Symfony 3.1 ainsi que sur le projet final avec la version 6.4.

Page Accueil - NON CONNECTE				Page Accueil - CONNECTE			
Versions	Total execution time	Symfony Initialization	Peak memory usage	Versions	Total execution time	Symfony Initialization	Peak memory usage
Symfony 3.1	1301ms	1107ms	25mb	Symfony 3.1	623ms	178ms	17mb
Symfony 6.4	39ms	12ms	4mb	Symfony 6.4	84ms	33ms	4mb

Liste des tâches - NON CONNECTE				Liste des tâches - CONNECTE			
Versions	Total execution time	Symfony Initialization	Peak memory usage	Versions	Total execution time	Symfony Initialization	Peak memory usage
Symfony 3.1	1968ms	216ms	17mb	Symfony 3.1	671ms	183ms	17mb
Symfony 6.4	138ms	36ms	4mb	Symfony 6.4	168ms	36ms	6mb

Liste des utilisateurs - CONNECTE			
Versions	Total execution time	Symfony Initialization	Peak memory usage
Symfony 3.1	662ms	200ms	17mb
Symfony 6.4	125ms	39ms	4mb

La mise à niveau du projet vers Symfony 6.4 ainsi que la version 8.3 de PHP a permis une réduction drastique du temps d'exécution des différentes pages de notre application avec une chute en moyenne de 86% du temps de chargement. Il en va de même pour l'occupation de la mémoire qui elle chute de près de 75% en moyenne.

Ces réductions sont significatives et prouvent la nécessité de la mise à niveau de notre application. Cette mise à jour réduit grandement le temps de chargement pour nos utilisateurs, ainsi que la tension serveur avec une charge réduite sur nos serveurs ce qui réduira notre empreinte énergétique ainsi que nos coûts serveur.

Pistes d'améliorations :

Les performances de notre application peuvent être encore améliorées. Prenons l'exemple de nos pages liste des tâches et des Users.

Actuellement, chaque visiteur qui charge l'une de ces pages déclenche des requêtes à la base de données pour récupérer les informations. Si nous avons des centaines ou des milliers de tâches ou d'utilisateurs, cela peut devenir coûteux en termes de temps d'exécution et de charge serveur.

La mise en place d'un système de mise en cache des requêtes permettrait d'améliorer grandement la situation. Ce système stockerait les résultats des requêtes pour une durée déterminée, de sorte que les utilisateurs suivants accédant à ces pages puissent bénéficier de données déjà récupérées et stockées en cache, éliminant ainsi le besoin de faire à nouveau appel à la base de donnée.

Modification du code (exemple fonction index du TaskController) :

```
#[Route('/tasks', name: 'task_list')]
16 references | 0 overrides
public function index(TaskRepository $taskRepository, CacheInterface $cache): Response
{
    $tasks = $cache->get('app.tasks', function (ItemInterface $item) use ($taskRepository) {
        $item->expiresAfter(3600); // Expire après une heure
        return $taskRepository->findBy([], ['id' => 'DESC']);
    });

    return $this->render('task/index.html.twig', ['tasks' => $tasks]);
}
```

Relevé profiler – page liste des tâches :

Liste des tâches - CONNECTE			
Versions	Total execution time	Symfony Initialization	Peak memory usage
Symfony 6.4 (sans cache)	168ms	36ms	4mb
Symfony 6.4 (avec cache)	78ms	15ms	4mb

Liste des tâches - CONNECTE		
Versions	Database queries	Query time
Symfony 6.4 (sans cache)	14	12ms
Symfony 6.4 (avec cache)	1	1ms

Relevé profiler – page liste des utilisateurs :

Liste des utilisateurs - CONNECTE			
Versions	Total execution time	Symfony Initialization	Peak memory usage
Symfony 6.4 (sans cache)	125ms	39ms	4mb
Symfony 6.4 (avec cache)	56ms	14ms	4mb

Liste des utilisateurs - CONNECTE		
Versions	Database queries	Query time
Symfony 6.4 (sans cache)	2	1,90ms
Symfony 6.4 (avec cache)	1	0,65ms

Des tests ont été faits en rajoutant cette logique de cache sur les fonctions index des «TaskController » et « UserController ». Nous pouvons donc voir encore une nette amélioration des performances de l'application. Le temps de chargement de nos pages réduit de plus de moitié. Le nombre de requêtes pour les tâches est drastiquement réduit ainsi que la durée des requêtes.

La mise en place de ce système de cache permettra un affichage encore plus rapide de nos pages. Mais surtout il assurera la scalabilité de l'application en nous préparant à l'arrivée d'un flux important d'informations (tâches, utilisateurs). Il faudra cependant veiller à mettre en place les logiques d'invalidité du cache pour les créations, modifications, suppressions des informations.