Rapport d'audit de l'application : Liste de Tâches

Prénom Nom

18 septembre 2024

Table des matières

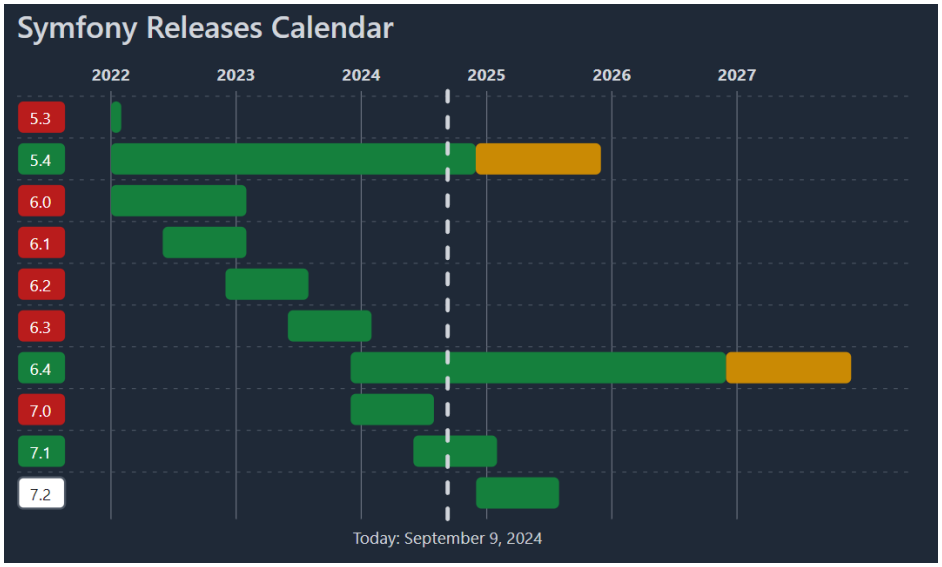
1 Introduction

Dans le cadre de mes études chez OpenClassrooms, j'ai été chargé d'auditer une application existante de gestion de tâches. L'objectif est de faire un point sur l'état actuel de l'application, tant du côté technique que du point

de vue de l'utilisateur, afin de proposer des améliorations et d'optimiser ses performances.

2 Migration vers une version plus récente de Symfony

L'application initiale utilisait Symfony en version 3.1. Pour bénéficier des dernières améliorations et assurer une meilleure maintenance, j'ai migré le projet vers Symfony 6.4.7, qui est la version avec support à long terme (`https://symfony.com/releases`).



Des ajustements ont été nécessaires pour remplacer les fonctionnalités dépréciées et adapter le code aux nouvelles conventions de Symfony.

Le projet nécessite au minimum PHP 8.1 pour fonctionner. J'ai utilisé PHP 8.1.14, et la base de données est en MySQL 5.7.36.

3 Anomalies à corriger

Plusieurs fonctionnalités devaient être améliorées :

— Les tâches doivent être automatiquement associées à l'utilisateur qui les crée.

— Lors de la modification d'une tâche, l'auteur ne peut pas être changé.

— Les tâches déjà existantes sans auteur doivent être rattachées à un utilisateur "anonyme".
— Ajouter la possibilité de choisir le rôle (utilisateur ou administrateur) lors de la création d'un utilisateur.
— Gestion des droits :
  — Seuls les administrateurs peuvent accéder aux pages de gestion des utilisateurs.
  — Un utilisateur ne peut supprimer que les tâches qu'il a créées.
  — Seuls les administrateurs peuvent supprimer les tâches de l'utilisateur "anonyme".

4 Problèmes rencontrés et solutions apportées

Pendant la migration, j'ai rencontré plusieurs difficultés :

— Les librairies utilisées n'étaient pas à jour. J'ai effectué une mise à jour via Composer.
— Le code devait être adapté aux nouvelles syntaxes et bonnes pratiques de Symfony 6.

Voici les changements principaux :

— La structure du projet a été modifiée pour correspondre à celle de Symfony 6.4.7.
— Les annotations dans les contrôleurs ont été mises à jour.
— La validation des formulaires utilise maintenant `isSubmitted()` en plus de `isValid()`.

L'ancienne version ne gérait pas les restrictions d'accès selon les rôles des utilisateurs, ni le cas des tâches sans utilisateur.

5 Fonctionnalités ajoutées

Pour améliorer l'expérience utilisateur, j'ai apporté quelques modifications supplémentaires :

— Ajout d'une barre de navigation affichant les liens disponibles en fonction du statut de l'utilisateur.
— Les tâches sont maintenant séparées en deux pages : tâches à faire et tâches terminées.
— Mise à jour de Bootstrap pour une légère refonte graphique.

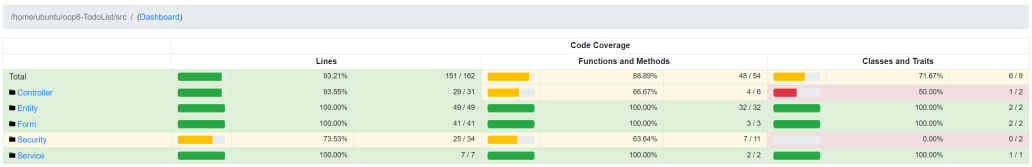
6 Tests unitaires et fonctionnels

J'ai mis en place des tests pour vérifier le bon fonctionnement de l'application :

— Tests unitaires pour vérifier la validité des entités et des contrôleurs.

— Création d'une base de données de test avec des données factices.

Les tests ont été réalisés avec PHPUnit.



/home/ubuntu/ocp8-TodoList/src / (Dashboard)

	Code Coverage					
	Lines		Functions and Methods		Classes and Traits	
Total	93.21%	151 / 162	88.89%	48 / 54	71.67%	6 / 9
Controller	93.55%	29 / 31	66.67%	4 / 6	50.00%	1 / 2
Entity	100.00%	49 / 49	100.00%	32 / 32	100.00%	2 / 2
Form	100.00%	41 / 41	100.00%	3 / 3	100.00%	2 / 2
Security	73.53%	25 / 34	63.64%	7 / 11	0.00%	0 / 2
Service	100.00%	7 / 7	100.00%	2 / 2	100.00%	1 / 1

7 Qualité du code

J'ai utilisé l'outil Codacy pour analyser la qualité du code et identifier les parties à améliorer.

`https://app.codacy.com/gh/MonNomUtilisateur/mon-projet/dashboard`

Codacy m'a aidé à supprimer le code inutile et à respecter les standards de codage. Le projet a obtenu une note de B.

8 Performance

8.1 Optimisation actuelle

Pour améliorer les performances, j'ai utilisé la commande suivante pour optimiser le chargement des classes :

```
composer dump-autoload --no-dev --classmap-authoritative
```

8.2 Résultats obtenus

Voici les résultats avant et après l'optimisation :

8.2.1 Avant optimisation

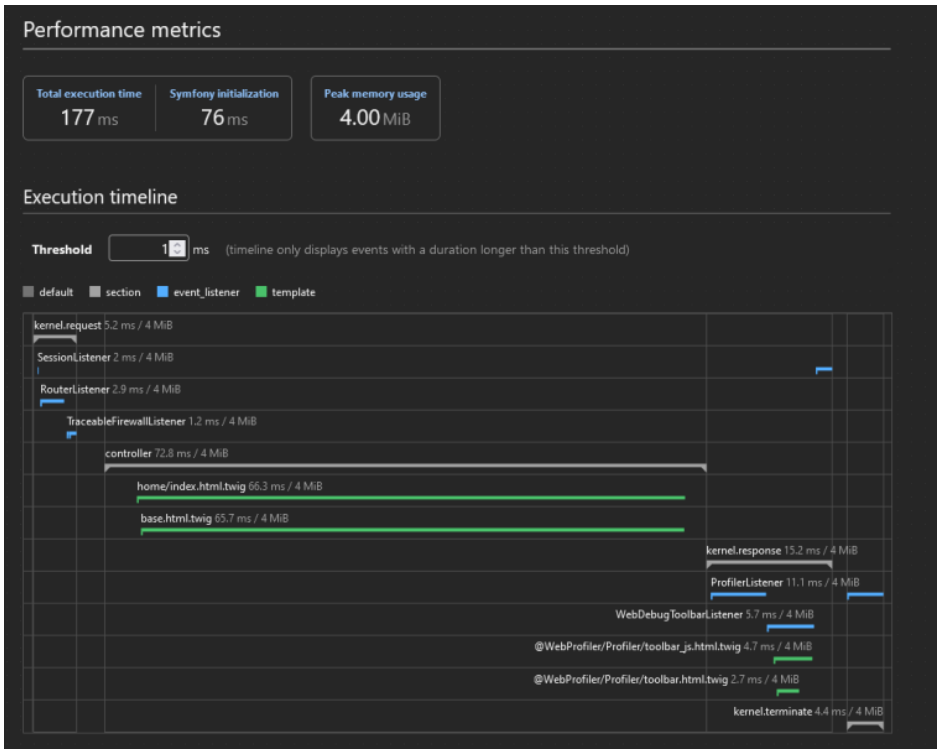


FIGURE 1 – Performances avant optimisation

8.2.2 Après optimisation

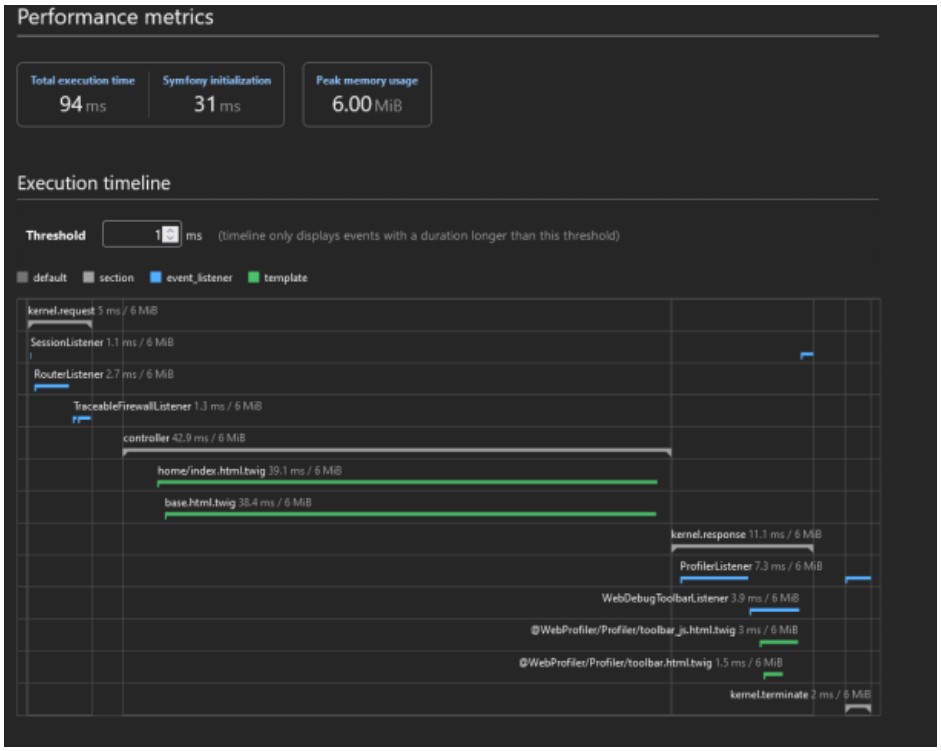


FIGURE 2 – Performances après optimisation

8.3 Pistes pour les futures optimisations

Pour aller plus loin, voici quelques pistes :

— Optimiser les requêtes vers la base de données.
— Mettre en cache les données fréquemment utilisées.
— Réduire la taille des fichiers CSS et JavaScript.
— Utiliser un CDN pour les ressources statiques.
— Optimiser les images et les médias.

```
# config/services.yaml
parameters:
  # ...
  .container.dumper.inline_factories: true
```

Le paramètre `.container.dumper.inline_factories` est utilisé lors de la compilation du conteneur.