Audit “Todolist”

Réalisé pour le projet 08 de la formation Développeur d’application PHP / Symfony de Openclassrooms.

1) Contexte

L’entreprise ToDo & Co m’a embauché pour améliorer la qualité de leur application.
Les demandes du clients sont:
1 Corrections des anomalies

- Une tâche doit être rattachée à un utilisateur
- Attribution d’un rôle lors de la création d’un utilisateur (et modification)

2 Implémentations

- Autorisations/ sécurité sur certaines pages et méthodes
- Test automatisés et ‘coverage’ du code

3 Création d’un document technique pour expliquer la mise en place et la maintenance de l’implémentation des autorisations

Les outils qui seront utilisés sont:

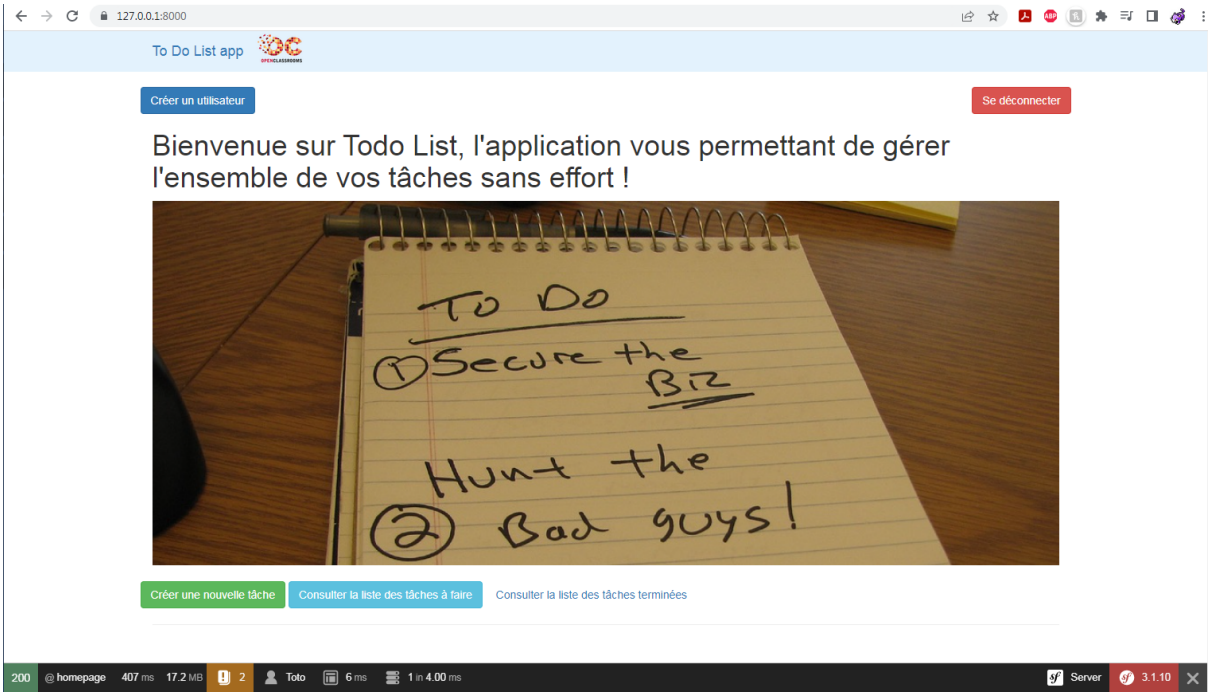
- Amélioration la qualité du code(Codacy)
- Amélioration de la performance du code (Symfony Insight)
- Résolution des éventuels bugs présents (tests exploratoires)
- Implémentation de nouvelles fonctionnalités (VSCode)
- Implémentation des tests (phpunit)
- Analyse et code coverage (xdebug)

Le projet est un site dont le code à été cloné à partir de ce repository github :
https://github.com/saro0h/projet8-TodoList
Il s’agit d’une application web de création de post-it / tâches et de pouvoir les marquer comme terminée.

2) Analyse du projet

2.a Description technique

- Le projet originel est en PHP : 5.5.9 et Symfony : 3.1.*
- Il semble contenir plusieurs bugs, notamment concernant les redirections et des morceaux de code pas encore implémentés
- Il y a peut de sécurité, les rôles ne sont pas bien définis et les accès admin ne sont pas protégés
- l'UI est un peu confuse pour un utilisateur et la structure du site n'est pas claire



ci dessus la page d'accueil du site lorsqu'un utilisateur est connecté

#Add more precise example of wrong code or practice

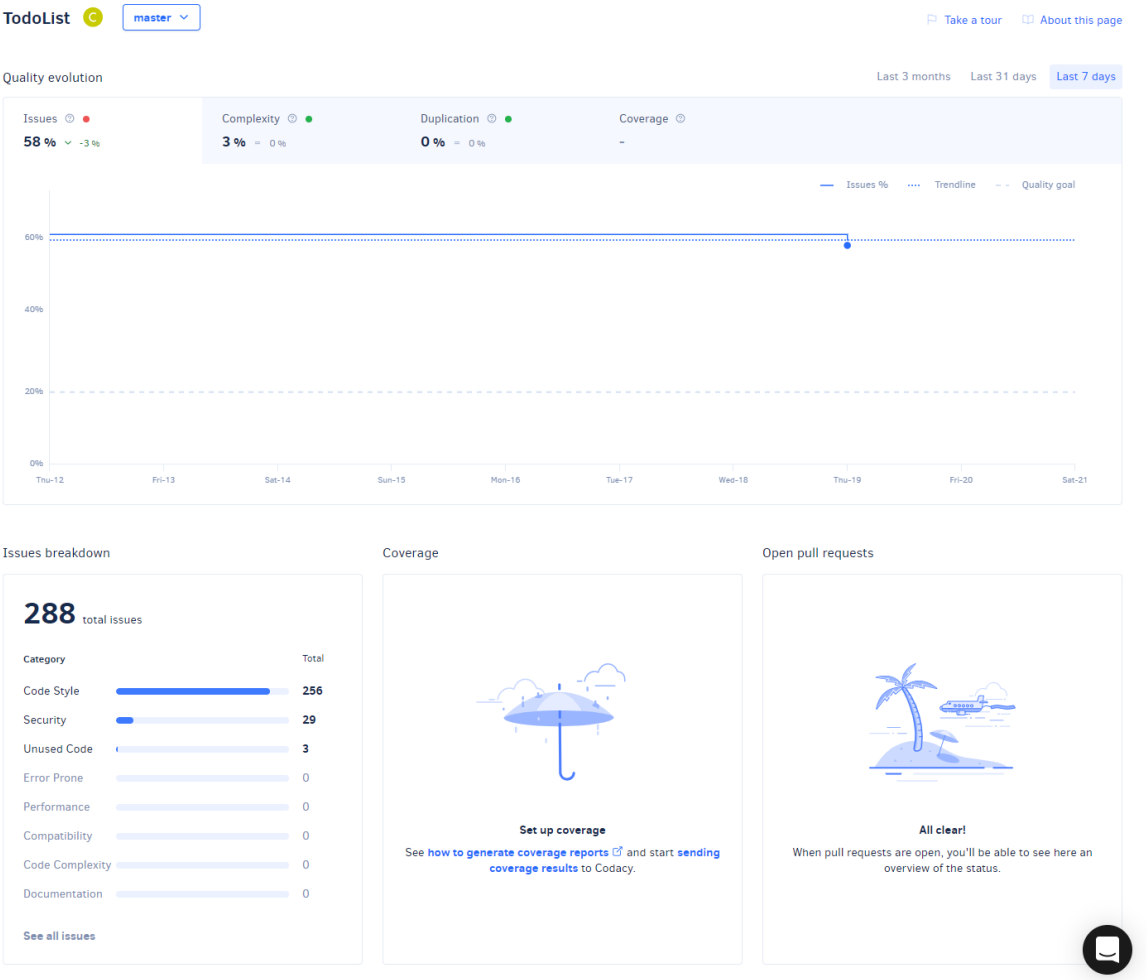
2.b Analyse du code

L'analyse du projet sur Codacy (ci-dessous) nous montre que le code obtient un score général de C et qu'il y a beaucoup d'issues relevées sur le repository.
Ces erreurs sont surtout :

- des lignes en doublon

- du code inutilisé
- des utilisations non recommandées de certaines méthodes
- un nommage des variables et indentations du code mal choisi
- etc…

Le nombre d'erreurs sur codacy n’est pas forcément réflecteur de la qualité du code mais nous donne un bon indice lorsqu'on regarde de plus prêt les raisons de ces erreurs. Nous utiliserons surtout le score général.



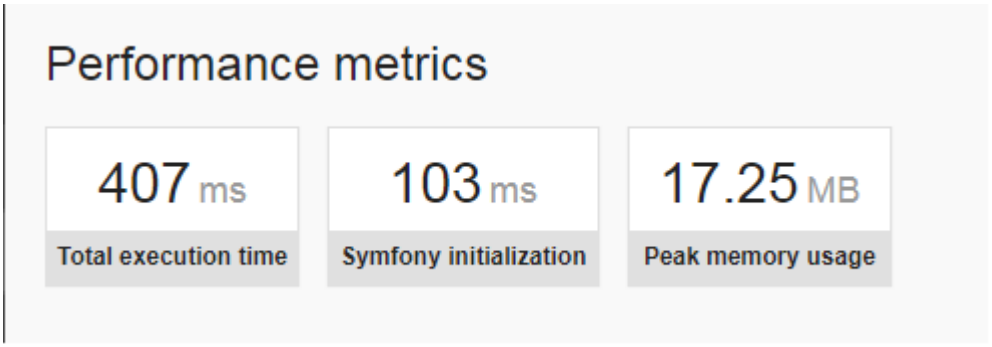
2.c Performances

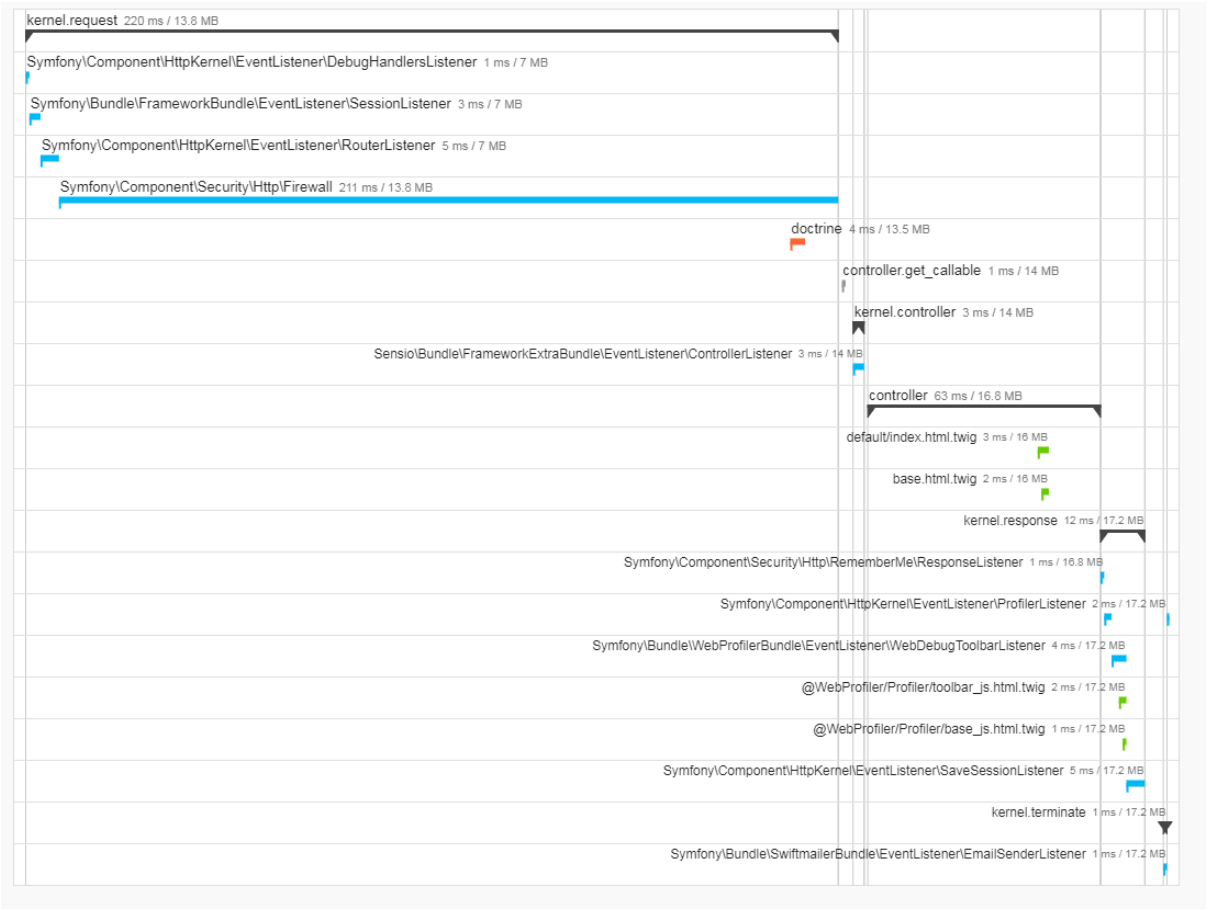
Symfony Insight nous donne des informations sur la performance de certaines routes, ainsi que leur life-cycle (le temps d'exécution des différents appels pour chaque route).
Avec les exemples ci-dessous, nous pouvons constater que certains ‘process’ prennent beaucoup de temps. Cela est dû pour plusieurs raisons:

- Le système de cache n’est pas optimisé
- Certaines fonctionnalités ne sont pas bien configurées
- Il y a des exécutions qui ne sont pas forcément utiles et qui sont sûrement due à une mauvaise base d’architecture du code.

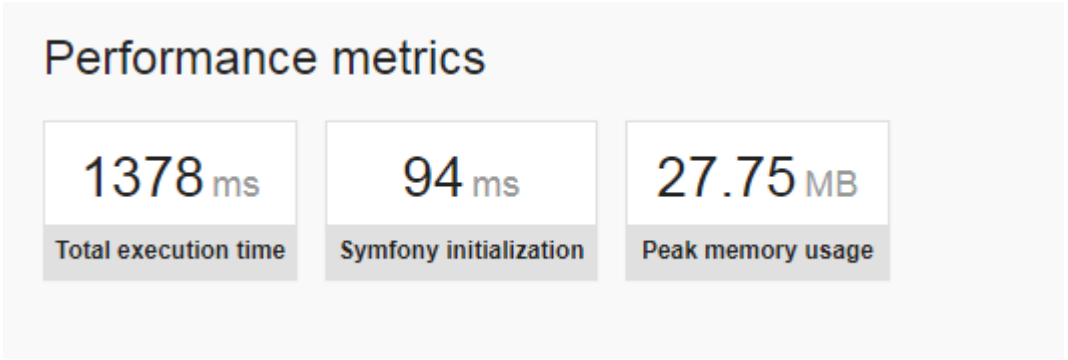
Ci dessous des exemples:

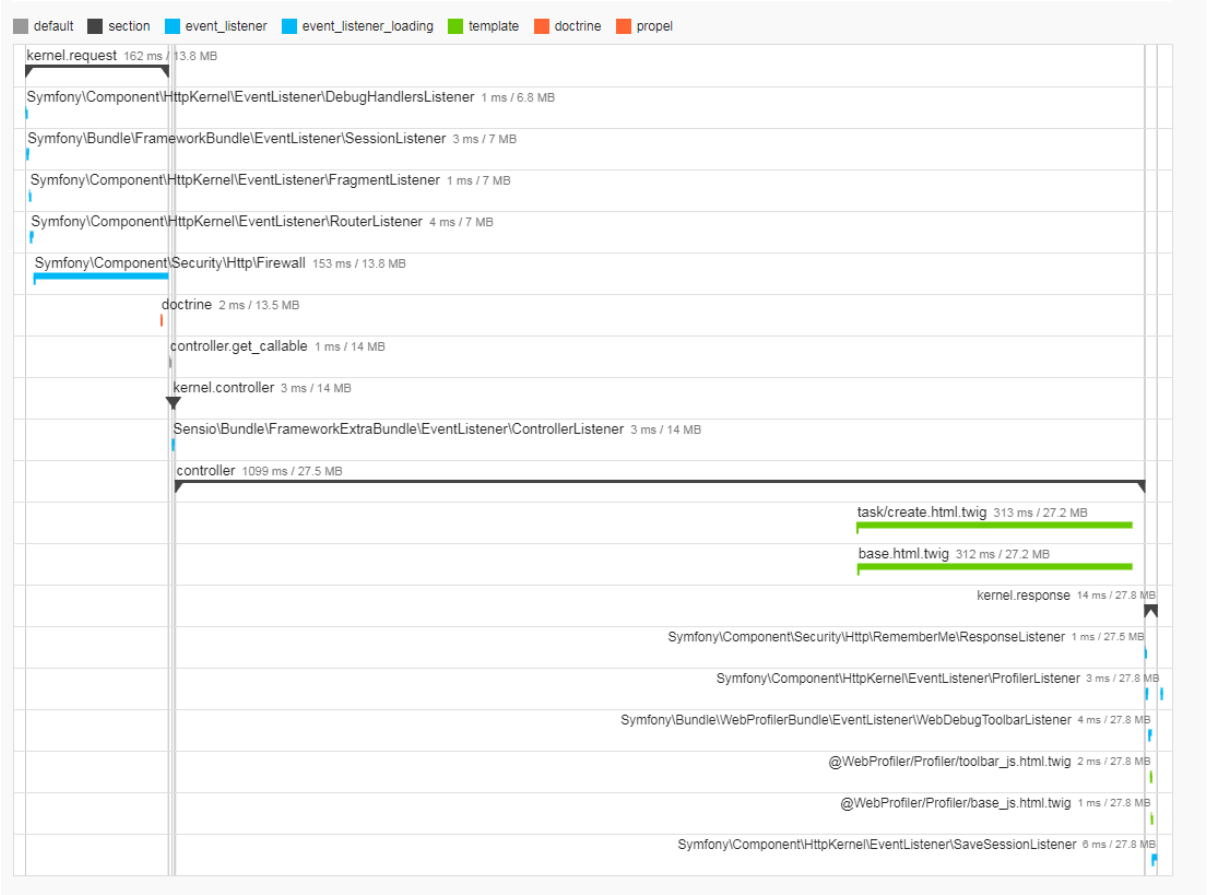
/home (affichage de la page d’accueil)



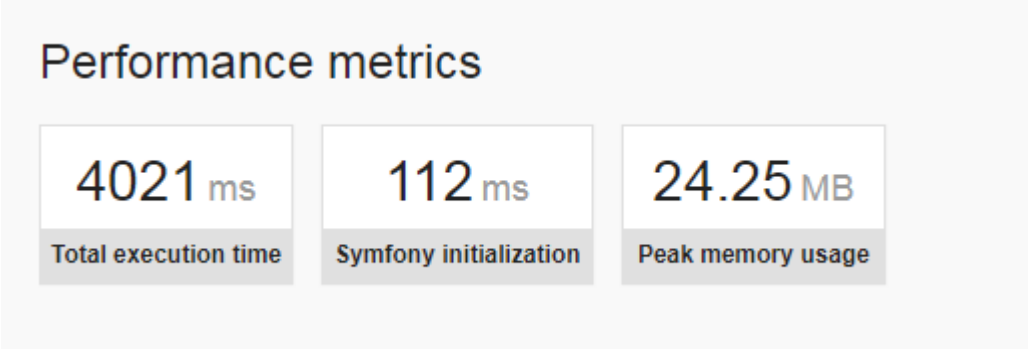


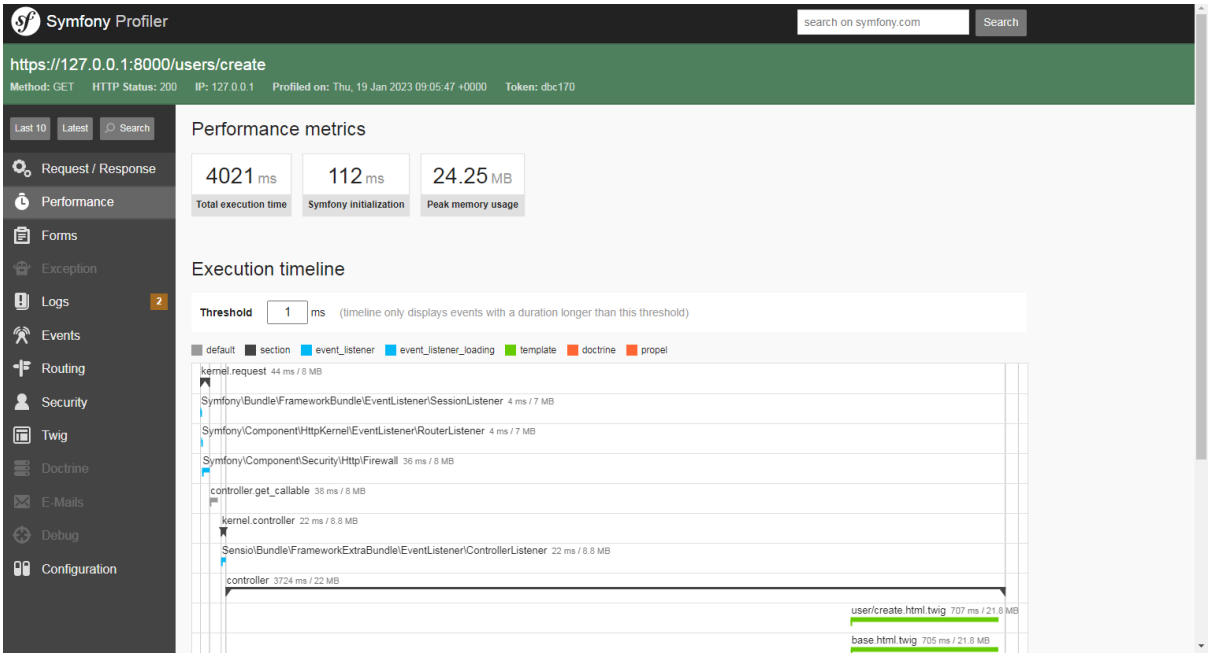
/task/create (création d’une tâche pour un utilisateur connecté)





/user/create (création d’un utilisateur en tant qu’admin)





2.d problèmes

En l'état, le projet comporte quelques problèmes et bugs dans le code:

- La Sécurité: Les différents rôles ne sont pas mis en place et les pages de créations et modification des utilisateurs n'ont pas de sécurité d'accès.
Le code étant dans une version si ancienne veux aussi dire que toutes les fonctionnalités liées à la sécurité ne sont plus mises à jour.

- L'organisation du code ne se base pas sur une organisation moderne et complique de ce fait l'exploration et le débogage par un développeur. De plus, certaines exécutions du code ne sont pas claires et difficilement contrôlables.

- Les bugs: Les redirections ne sont pas à jour et l'organisation du code mène à des comportements non prévisibles et source de bug.

- L'absence de test: N'ayant pas de tests automatiques, l'application ne permet pas un contrôle de l'ensemble des routes sauf avec des tests d'explorations.

3) Première version

3 a) Direction

Dans cette première partie, le focus était placé sur la correction des bugs, la migration vers une version plus récente des langages et la mise en place des différentes fonctionnalités demandées.
Cette version, bien que plus performante et plus claire, comporte beaucoup de problèmes développés dans la section suivante.
La version est disponible sur le repository git: https://github.com/CHBHR/TodoList

3 b) Évolutions et problèmes rencontrés

Dans cette section je décrirai les évolutions du projet et les problèmes éventuellements rencontré par groupe:

⇒La version de PHP a été mise à jour à PHP 7.4.26 grâce notamment au fichier .php-version et la version de Symfony à 4.4 grâce au composer.json.
Problème: Le contrôle des versions des librairies (dont certaines n'étant plus maintenues) a rendu les migrations beaucoup plus complexes et certains fichiers comme le kernel.php ou la plupart des fichiers .yml auraient dû être modifiés, renommés ou supprimés car obsolètes.

⇒ Les bugs connus et découverts ont été résolus.
Problème: La migration a rendu certaines fonctionnalités et implémentations de classe dépréciées et l'implémentation de nouvelles classes ou la création de classes abstraites est devenue nécessaire.
De plus, le fichier composer.lock a lui aussi commencé à causer des issues dues à sa complexité.

⇒ La sécurité à été améliorée grâce à l'implémentation de Rôle et de droit dans le Firewall de Symfony pour gérer les droits administratifs de certaines routes.
Problème: Le Firewall a dû être mis à jour sur une version de Symfony qui n'est plus maintenue, créant ainsi des problèmes liés à la sécurité et complexifiant le processus. De plus, les systèmes de sécurité de classe n'étant pas à jour représentaient un risque élevé de la sécurité du projet.

4) Décision et nouvelle version

4 a) Décision

Après plusieurs semaines à essayer de régler les différents problèmes et après discussion avec des collègues de travail, il a été décidé d'essayer de recréer le projet 'from scratch' dans une version à jour de Symfony pour plusieurs raisons:

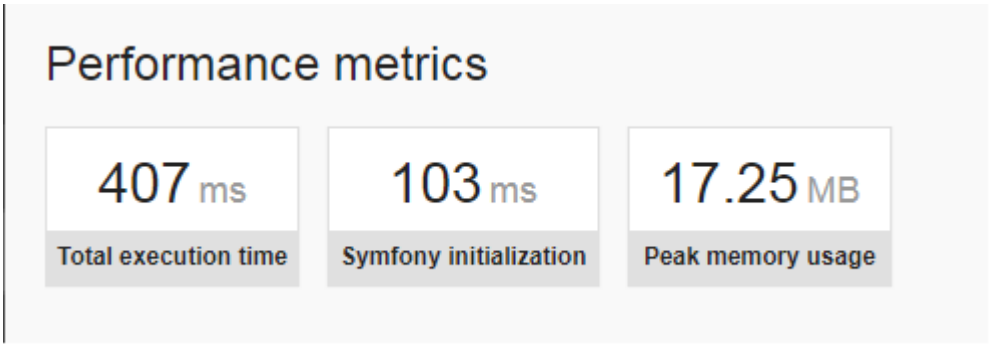
- La génération d'un nouveau projet Symfony en ligne de commande nous permet d'avoir une architecture à jour et dans les normes de ce qu'il se fait aujourd'hui. Permettant une meilleure organisation du code, de meilleures performances et plus de facilité pour les futurs développeurs du projet pour ajouter des fonctionnalités, gérer les éventuels bug ou tout simplement se référer à la documentation officielle.
- Les librairies utilisées sont compatibles directement avec les dernières versions de PHP et Symfony, donc stables et plus sécurisées. Les fichier composer sont donc plus simple à parcourir, clair et moins 'lourd' à exécuter par notre site.
- Les fichiers de configuration ont été allégés et l'exécution du code est beaucoup plus simple à comprendre. On évite ainsi les problèmes liés aux 'processus' non documentés des versions du projet précédentes et les baisses de performances.
- Le temps: La mise en place du projet sur de nouvelles bases a été rapide à mettre en place et sera par la suite plus rapide à gérer / debugger par les développeurs. Après avoir passé plusieures semaines à régler des bugs de migration sur la version précédente, il a semblé évident que le mieux pour la suite était de faire une nouvelle version du projet.

5) Performances

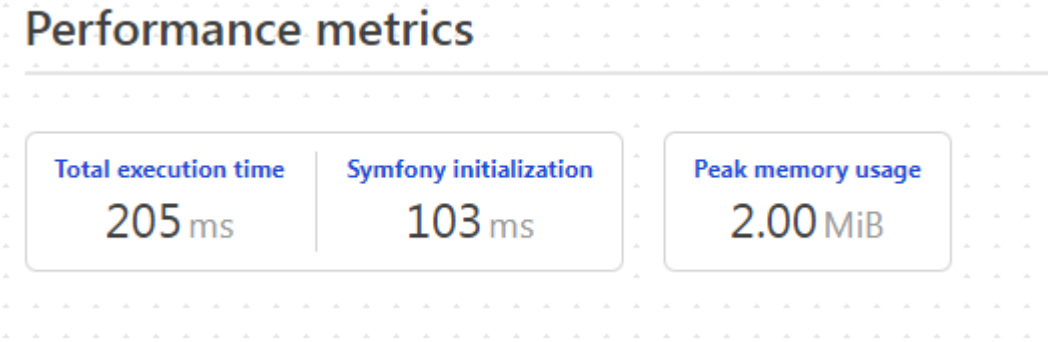
ToDoList_Remastered est la nouvelle version du projet ToDoList codée en PHP 8.1.0 et Symfony 6.2.7.
Le repository est disponible ici: https://github.com/CHBHR/TodoList_Remastered

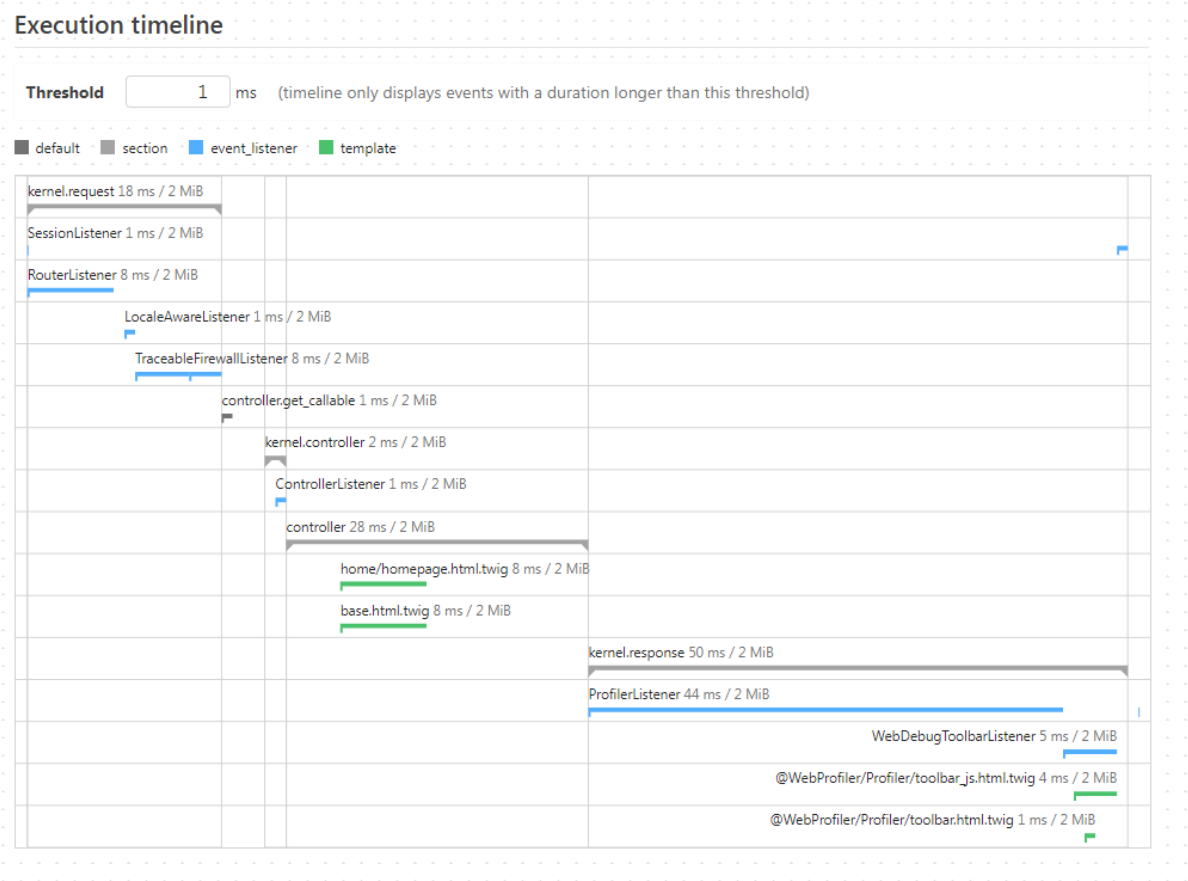
/home

ancienne version



nouvelle version

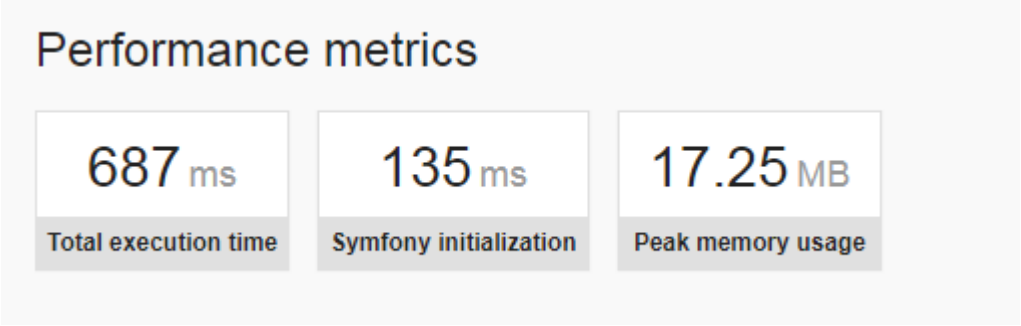




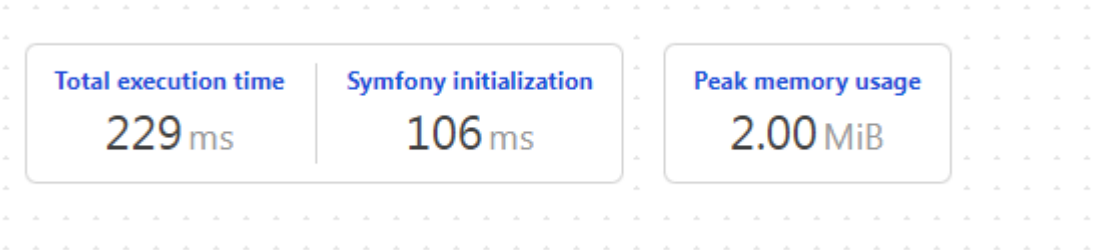
Ici on peut voir que la performance lors de l'affichage de la page d'accueil est nettement meilleure que la version initiale et l'utilisation de la mémoire a été optimisée.

/login

ancienne version

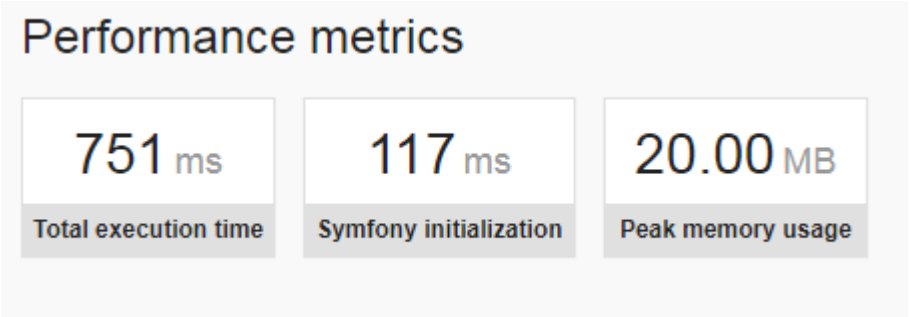


Nouvelle version

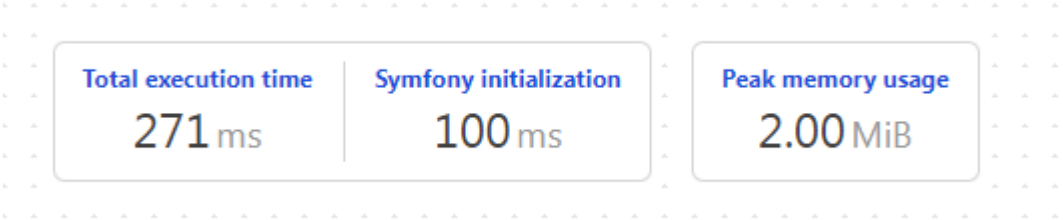


/admin/users

ancienne version



nouvelle version



Front-end

En plus d'avoir améliorer les performances backend, l'UI a aussi été repensé pour permettre une meilleure expérience utilisateur et de meilleures performances.

L'extension "LightHouse" de google chrome nous permet de faire un benchmark de notre application et nous faire remarquer que le front est également performant et améliorable. (voir img ci-dessous).

Le référencement (SEO) pourrait être amélioré dans le futur pour que le site soit plus visible dans les résultats de recherche.

https://127.0.0.1:8000/home

83	93	92	67	PWA
Performance	Accessibility	Best Practices	SEO	PWA

83

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS Expand view

■ First Contentful Paint 2.3 s	● Time to Interactive 2.3 s
● Speed Index 3.4 s	● Total Blocking Time 0 ms
● Largest Contentful Paint 2.5 s	▲ Cumulative Layout Shift 0.357