AUDIT D'APPLICATION

L'audi de qualité du code et de performance à été realisé aprés la correction d'anomalie et l'ajout de nouvelle fonctionnalités (minum viable product). Aprés un etats des lieux global je suis à même de proposer un plan d'amélioration correspondant à une utilisation courante de l'application ToDo & Co que nous allons vori ci-dessous.

1. Qualités du code

1A . Respect des normes :

Le code php respect les normes PSR-2, il respect donc, les normes strictes en terme de nommage, d'indentation et d'espeacement. il est commenté avec PHPDoc et le répertoire GitHub où a été déposé le projet, aprés avoir subi une revue de code automatisé par codacy, a obtenu le badge 1 qui certifie la bonne qualité du code.

Amélioration possible :

la version actuelle de l'application symfony est 3.2 elle n'est plus maintenue par Sensiolabs donc, une migration vers la version la plus récente est recommandé afin de bénéficier d'un support longue durée avec des corrections contre les failles de sécurité. Je n'ai pas voulu update sur une version minimal 4.4 car, trés prochainement il y aura une nouvelle LTS et je ne veux pas que le projet soit déprésié à nouveau dans quelques semaines.

2. Les performances

pour mesurer la performance de l'application j'utilise blackfire.io il ne requiert aucune modification du code et permet de trouver les causes de probléme de performance.

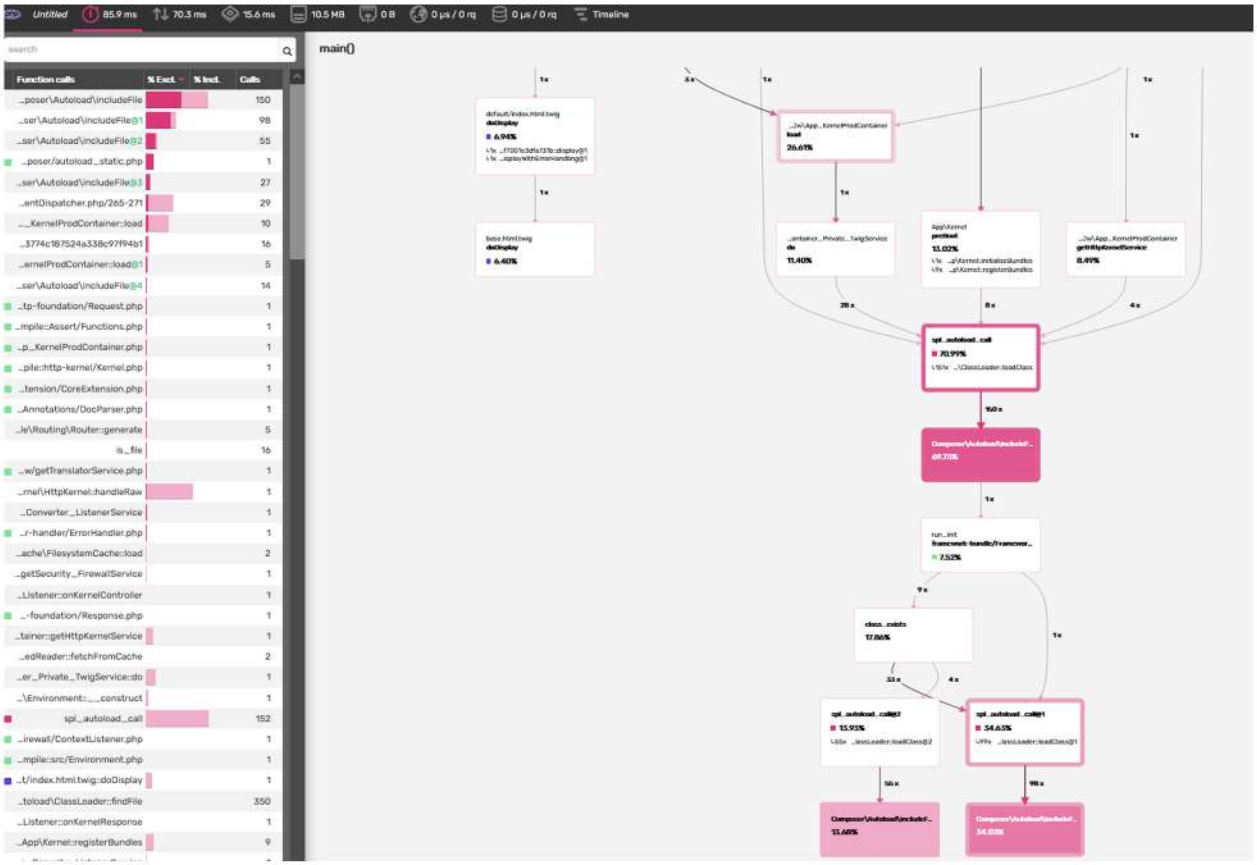
Performance global :

avant de commencer il faut s'avoir que les données collectées sont tributaires des performance de la machine (type de memoire vive, processeur, disque dur etc ...). mais, nous pouvons affirmer que les performance de l'application sont cohérente et surtout homogénes et ne semblent pas dépasser les mesure de performance optimal ce qui est un bon point. ainsi le temps d'exécution moyen d'une route est de 263 ms donc, en dessous des 1s.

Route	Méthode HTTP	Temps d'exécution - ms	Mémoire - mb
/	GET	85.9	10.5
/login	GET	111	13.1
/login_check	POST	539	14.3
/logout	GET	141	13.6
/tasks	GET	239	17.8
/tasks/create	GET	240	27.3
/tasks/create	POST	432	21.6
/tasks/{id}/edit	GET	204	21.4
/tasks/{id}/edit	POST	200	19.2
/tasks/{id}/toggle	GET	126	14.5
/tasks/{id}/delete	GET	125	14.5
/users	GET	163	17.6
/users/create	GET	250	21.9
/users/create	POST	540	21.8
/users/{id}/edit	GET	220	22
/users/{id}/edit	POST	593	20.1

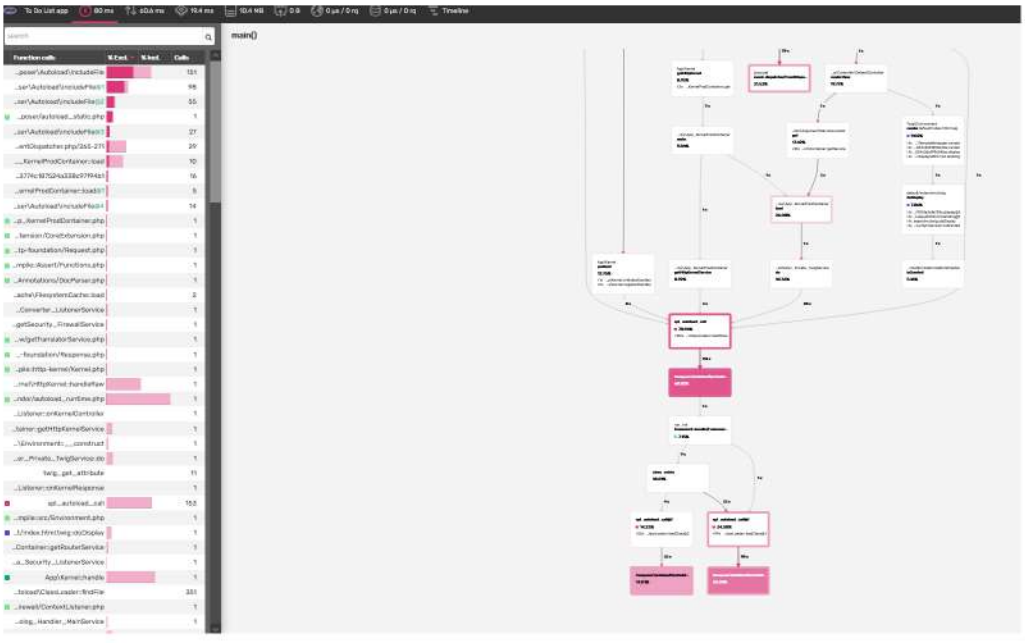
Il faut s'avoir que les données collectées sont tributaires de la memoire vive, du processeur et du disque dur de l'ordinateur qui a généré le resultat, et qui eux mêmes fonctionnent localement avec le service de symfony serve. Mais sur l'ensemble des pages, on peut affirmer que les résultat obtenus sont homogénes, et ne semblent pas dépasser la mesure qu'on attends d'une application performante. Ainsi, le temps d'éxecution moyen d'une route est de 263ms , soit largement en dessous du baréme de 1 seconde.

Voici dans le détail le profil de la page home :



Le bloc de gauche donne la liste de toutes les fonctions executée par notre script php. Les plus lentes en fonction du temps d'éxécution sont en haut de la list. quant au chemin en surbrillance, à droite, il montre les fonctions dles moins performantes et aide à choisir lesquelles doivent être optimiser.comme pour l'ensemble des pages, on note que la fonction la plus couteuse en ressource est loaderclass qui integrete file_exists. elles sont appelées depuis l'autoload de composer et sont natives au framework symfony, heureusement, il est possible d'optimiser ce chargement avec la commande composer dump-autoload -o qui évite la vérification systématique de fichier dans l'autoloader de composer

Voici le resultat :



Nous avons gagné pas moins de 5.9 ms ce qui correct, par ailleurs nous pourrions encore améliorer les statistiques par l'update de jquery et de bootstrap. le listing des amelioration possible si dessous nous donnera des meilleurs performances et expérience utilisateur.

3. Les Ameliorations suggérez

Amélioration des performances :

- Migration vers derniére version de symfony.
- Mise en cache de composer.
- Migration vers nouvelle version de bootstrap voir utiliser un nouveau systéme de templating comme Tailwind css qui est bien plus performant.
- Migration vers nouvelle version de Jquery ou faire les requete ajax en vannila JS voir pour plus de fluidité utilisé vue.js + socket.io.
- Changer la structure dans la base de donnée au niveau du created_at dans task en auto_increment.

Amélioration expérience utilisateur :

- Ajout d'un bouton de suppresion d'utilisateur dans la consultation de la liste des utilisateurs.
- Ajout d'une category pour les tâches.
- Ajout d'une priorité pour les tâches.
- Ajout de vue.js et socket.io pour plus de fuidité dans l'application.
- Modifier le rendu des vues pour les rendres un peu moins flat pour ce démarquer d'autres applications.
- Ajout d'un systeme de bbcode dans la création des tâches
- Ajout d'un nouveau favicon moins générique