ToDo & Co

Audit de qualité et performance du code

Version initiale : Symfony **3.1** & PHP **5.6**
Version mise à jour : Symfony **6.2** & PHP **8.2**

Mise à jour de l'application

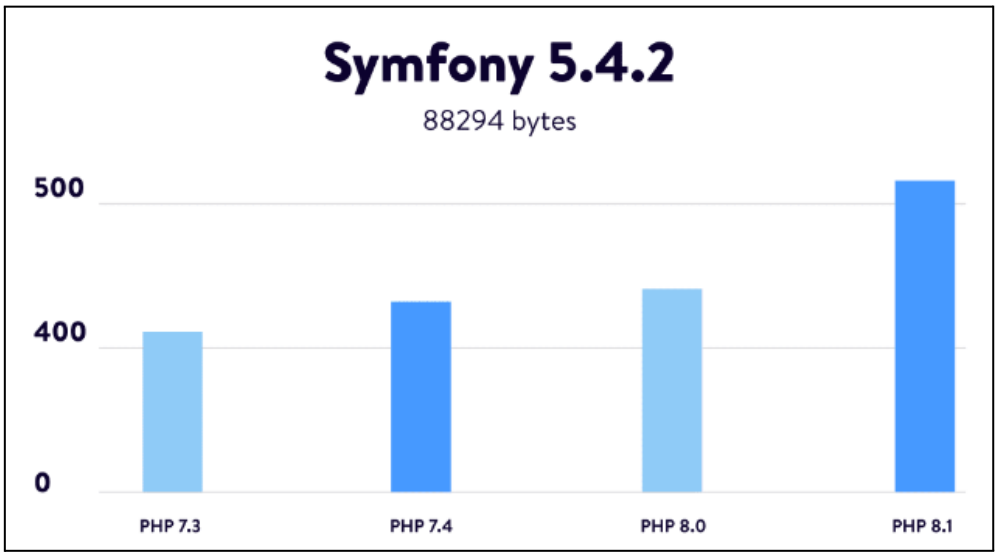
En plus des corrections effectuées et des implémentations de nouvelles fonctionnalités, l'application a été mise à jour de **Symfony** 3.1 à 6.2.

Cette mise à jour de Symfony s'accompagne d'une montée de version **PHP** 5.6 à 8.2.

Il est important de noter ici qu'une montée de versions Symfony & PHP s'accompagne souvent d'un **gain** en termes de **performances**.

Par exemple, le site **Kinsta.com** a effectué des benchmarks sur Symfony 5.4.2 - utilisant le projet de démonstration du framework.

Les résultats, en termes de requêtes par seconde, sont les suivants :



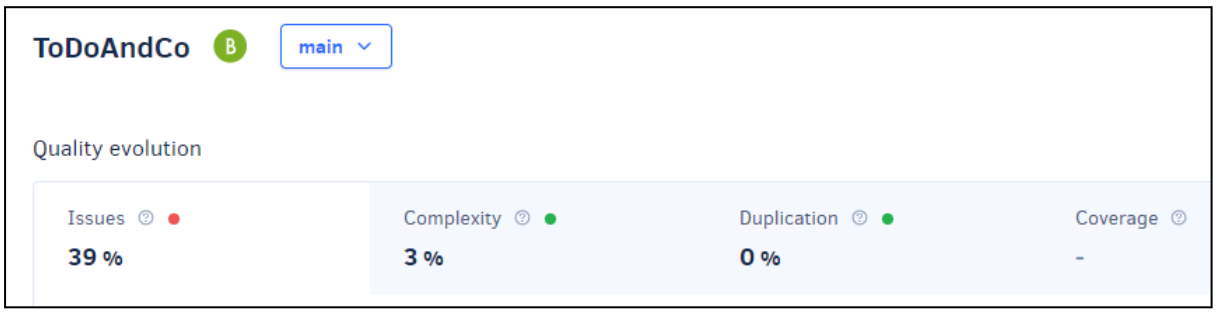
En termes de qualité du code, les mises à jour induisent aussi une meilleure lisibilité, grâce notamment aux **attributs** PHP 8 (qui remplacent le système d'annotations en commentaires) ou à la **structure** du projet, plus claire, en Symfony 6.

Qualité du code : version initiale

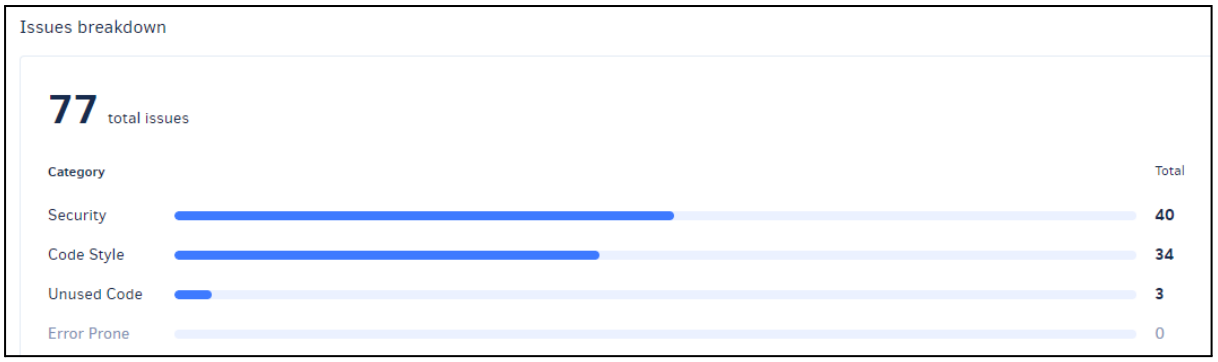
Une fois le projet récupéré et initialisé dans sa version initiale, **Codacy** a été utilisé pour faire une première passe d'analyse du code.

Cet outil nous permet de relever certains problèmes, divisés en plusieurs catégories :
- **Code Style**, **Security**, **Performance**, **Complexity**, et bien d'autres.

Le premier rapport de Codacy - sans aucun changement du code existant - nous donne le résultat suivant :



Les problèmes (*issues*) trouvés se décomposent comme suit :



On détecte alors un nombre assez élevé de problèmes potentiels au niveau **sécurité** et **style du code**.

Beaucoup de ces problèmes détectés vont être réglés au passage en Symfony 6 et PHP 8.

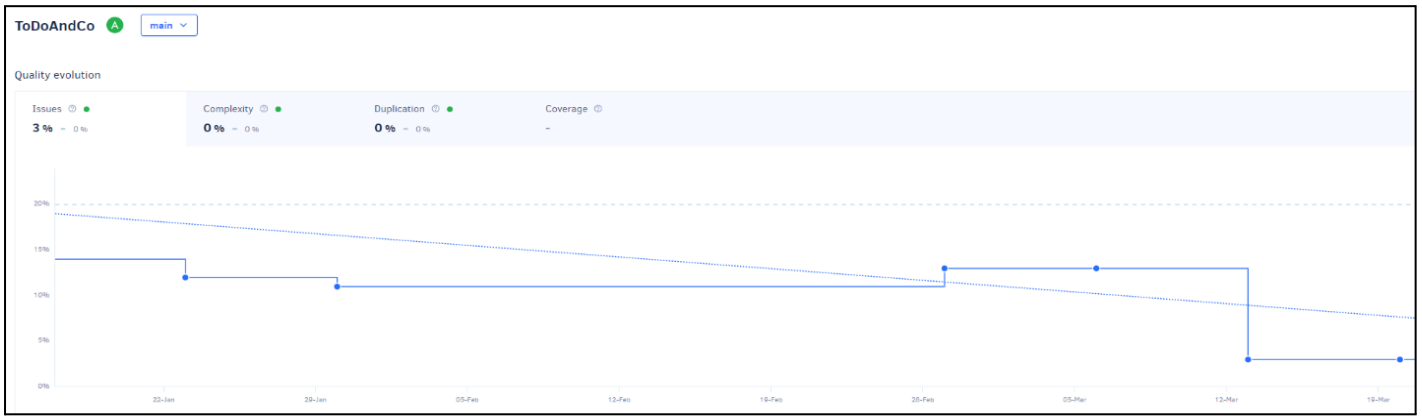
Qualité du code : nouvelle version

En mettant à jour le projet, et avant toute modification de fonctionnalités, une nouvelle passe de **Codacy** a été effectuée.

Voici la différence entre les deux versions du projet à cet instant :

Nous passons donc d'une analyse avec **39%** d'issues à **14%**. Les quelques alertes sur la **complexité** du code ont aussi disparu.

Enfin, une fois les **corrections** faites et les **fonctionnalités** développées, Codacy nous indique dans [son dernier rapport] cet historique :



Performances

Les mesures de performances sur les deux versions de l'application ont été réalisées avec la *toolbar* de Symfony.

La Toolbar Symfony nous permet d'avoir des informations sur les **requêtes** effectuées, l'utilisation de la **mémoire**, le temps de chargement total, le temps d'exécution détaillé, etc.

Optimisations

Dans la configuration de PHP, nous avons activé **OPCache**, une extension qui améliore les performances en sauvegardant un cache des scripts en mémoire.

Ceci permet d'éviter de charger et parser le code PHP à chaque requête.

Une autre optimisation potentielle serait d'utiliser le bundle **Webpack Encore**. Ce dernier permet, notamment, de regrouper tous les scripts JS ou fichiers CSS d'une application en un seul fichier.

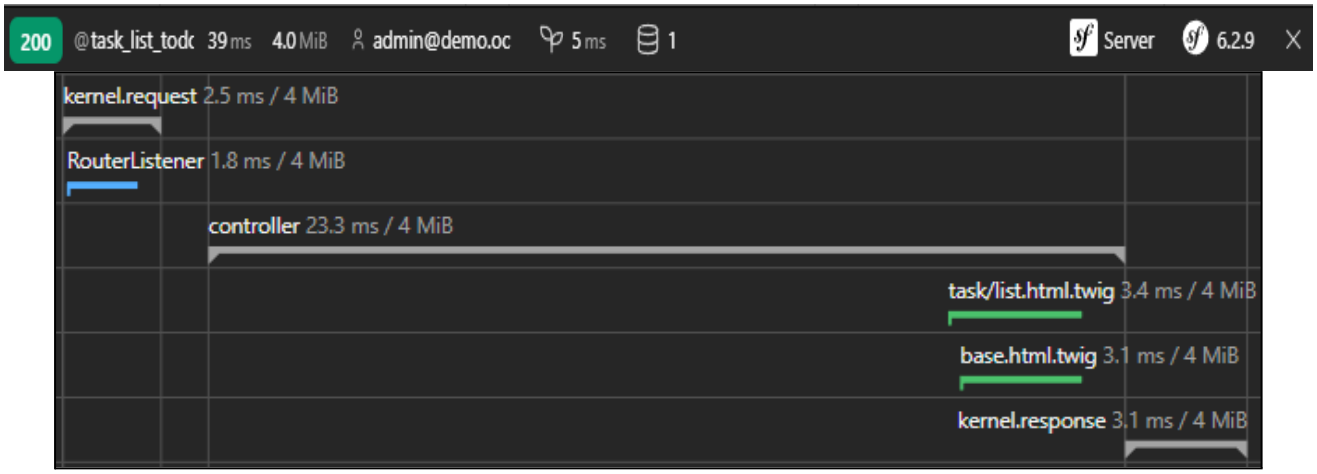
Se faisant, il réduit le nombre de requêtes HTTP effectuées et améliore les performances.

Encore Bundle est disponible pour Symfony et permet d'intégrer simplement et de configurer **Webpack**.

Comparaison de performances : Page de liste des tâches

- Version de base

- Version mise à jour



On peut alors constater plusieurs améliorations :

- La page se charge en **39** ms contre **247** ms en version initiale
- La mémoire utilisée passe à **4** mo contre **15** mo en version initiale
- Sur le framework : les temps d'exécutions du Controller et du Render des templates sont plus bas, et la partie firewall a été optimisé
    - La page est pourtant plus complexe à produire, car nous filtrons les droits d'opérations sur les tâches au moment de créer la vue

On retrouve des résultats similaires sur le reste des pages de l'application.