

# **Audit de qualité et de performance de code**

## **Application To Do List**



**Janvier 2024  
Marion Doubeck – ToDo & Co**

## Sommaire

### 1. Introduction

- Contexte de l'audit
- Objectifs et portée de l'audit

### 2. Méthodologie

- Outils utilisés : Codacy, Lighthouse, Symfony Profiler, PHPUnit
- Critères d'évaluation de la qualité et de la performance
- Périmètre de l'audit : code source, architecture, tests, performance

### 3. Justification de la refonte totale de l'application Symfony 3.1 en Symfony 6.4

- Obsolescence technologique
- Support et maintenance
- Performances et optimisations
- Moindre dette technologique
- Temps et efficacité
- Difficultés de déploiement avec Docker

### 4. Évaluation de la qualité du code

- Analyse avec Codacy : résultats et recommandations
- Tests unitaires et fonctionnels avec PHPUnit
- Couverture de code : 96.53%
- Conformité aux bonnes pratiques de développement Symfony

### 5. Évaluation de la performance de l'application

- Analyse avec Lighthouse : résultats et recommandations
- Utilisation du Symfony Profiler : optimisations possibles
- Mesures de performance : temps de chargement, requêtes SQL, utilisation des ressources

### 6. Conclusion

- Récapitulation des principales conclusions de l'audit
- Perspectives d'amélioration pour l'application Symfony

# **1. Introduction**

Le développement d'une application est une course effrénée où chaque ligne de code compte. Dans le contexte de notre startup émergente, ToDo & Co, l'urgence de démontrer la viabilité de notre produit auprès des investisseurs a conduit à la création rapide d'un Minimum Viable Product (MVP). Notre choix stratégique initial s'est porté sur Symfony, un framework PHP robuste et éprouvé, pour réaliser cet exploit dans les délais serrés.

Maintenant, alors que la startup a levé des fonds et que notre application a le potentiel de devenir une véritable référence dans la gestion des tâches quotidiennes, nous nous retrouvons à un tournant crucial. Mon rôle, en tant que nouveau développeur au sein de l'équipe, est de mener l'application vers de nouveaux sommets en améliorant sa qualité sous tous ses aspects.

Dans cette optique, ce projet de spécialisation s'articule autour de trois axes principaux : l'implémentation de nouvelles fonctionnalités, la correction d'anomalies existantes, et la mise en place de tests automatisés pour garantir la fiabilité du code. Plus précisément, nous nous attellerons à permettre la liaison entre les utilisateurs et les tâches, à offrir des rôles personnalisés pour une gestion fine des droits, et à sécuriser l'application avec une authentification robuste.

Cependant, le travail ne s'arrête pas là. Nous devons également documenter notre travail de manière claire et concise, afin de faciliter l'intégration de futurs développeurs et de préparer le terrain pour une collaboration efficace. De plus, l'audit de qualité du code et de performance de l'application que nous entreprenons permettra d'identifier les zones d'amélioration et de réduire la dette technique accumulée.

En somme, ce projet représente une opportunité unique de façonner l'avenir de ToDo & Co en élevant l'application à de nouveaux standards de qualité et de performance. Forts de nos compétences et de notre engagement, nous sommes prêts à relever ce défi avec détermination et excellence.

## 2. Méthodologie

La méthodologie adoptée pour mener à bien notre projet d'amélioration de l'application ToDo & Co repose sur l'utilisation d'outils et de pratiques bien définis, visant à garantir la qualité et la performance de notre travail.

**Outils utilisés** : Pour évaluer la qualité du code et la performance de l'application, nous avons utilisé plusieurs outils spécialisés. Codacy a été utilisé pour l'analyse statique du code, nous fournissant des indications précieuses sur les problèmes potentiels et les améliorations à apporter. Pour évaluer la performance de l'application, nous nous sommes appuyés sur le Symfony Profiler et google lighthouse. Des tests unitaires et fonctionnels permettent également de vérifier la qualité du code.

**Critères d'évaluation** : Nous avons défini des critères d'évaluation clairs pour mesurer la qualité et la performance de l'application. Pour la qualité du code, nous avons pris en compte des aspects tels que la lisibilité, la maintenabilité, et la conformité aux bonnes pratiques de développement Symfony. Pour la performance de l'application, nous avons évalué des métriques telles que le temps de chargement des pages, le nombre de requêtes SQL, et l'utilisation des ressources système.

**Périmètre de l'audit** : L'audit de qualité et de performance a porté sur plusieurs aspects de l'application, notamment le code source, l'architecture, les tests automatisés, et les performances globales de l'application. Nous avons veillé à couvrir l'ensemble des fonctionnalités critiques de l'application, ainsi que les points sensibles en termes de qualité et de performance.

### 3. Justification de la refonte totale de l'application

#### Symfony 3.1 en Symfony 6.4

Dans le cadre de la maintenance et de l'amélioration d'une vieille application Symfony 3.1, j'ai pris la décision de procéder à une refonte totale de l'application en utilisant la version la plus récente, Symfony 6.4. Cette décision a été motivée par plusieurs facteurs clés qui démontrent les avantages et la pertinence de cette approche.

**1. Obsolescence technologique** : La version 3.1 de Symfony, bien qu'elle ait été une version LTS (Long Term Support) à l'époque de sa sortie, est désormais obsolète. Avec l'avancement constant des technologies et les nouvelles fonctionnalités introduites dans les versions ultérieures de Symfony, rester sur une version aussi ancienne expose l'application à des risques de sécurité, des bugs non corrigés et une obsolescence progressive.

**2. Support et maintenance** : Le maintien d'une application sur une version obsolète de Symfony implique souvent des difficultés croissantes en termes de support et de maintenance. Les ressources disponibles pour aider à résoudre les problèmes rencontrés deviennent de plus en plus limitées au fil du temps, ce qui peut entraîner des retards dans les correctifs et des temps d'arrêt prolongés.

**3. Performances et optimisations** : Les versions plus récentes de Symfony sont généralement optimisées pour des performances accrues, avec des améliorations significatives dans le traitement des requêtes, la gestion de la mémoire et l'efficacité globale de l'application. En migrant vers Symfony 6.4, nous pouvons bénéficier de ces améliorations pour garantir une expérience utilisateur plus fluide et réactive.

**4. Moindre dette technologique** : En procédant à une refonte totale de l'application en Symfony 6.4, nous avons pu repartir sur des bases solides et modernes, réduisant ainsi la dette technologique accumulée au fil du temps. Cela nous permet de bénéficier des meilleures pratiques de développement, des normes de sécurité actuelles et des fonctionnalités avancées offertes par Symfony 6.4.

**5. Temps et efficacité** : Bien que l'idée de patcher l'application existante puisse sembler être une solution rapide à première vue, la mise en œuvre de correctifs sur une base obsolète peut souvent être plus laborieuse et chronophage que prévu. En optant pour une refonte totale en Symfony 6.4, nous avons pu réaliser les changements nécessaires de manière plus efficace et rapide, sans compromettre la qualité du code.

En conclusion, la décision de refondre totalement l'application en Symfony 6.4 plutôt que de faire des patchs correctifs sur la version 3.1 repose sur une analyse approfondie des risques, des avantages et des opportunités. Cette approche permet non seulement de garantir la pérennité et la sécurité de l'application, mais aussi de tirer pleinement parti des fonctionnalités et des performances offertes par la dernière version de Symfony.

**6. Difficultés de déploiement avec Docker** : Une autre considération importante dans la décision de refonte totale de l'application en Symfony 6.4 a été les difficultés rencontrées pour faire tourner la version 3.1 de Symfony avec la version actuelle de Docker. La gestion des dépendances et des configurations dans un environnement Dockerisé peut devenir complexe, surtout lorsqu'il s'agit de versions obsolètes de frameworks et d'outils. Dans ce cas précis, la version de Docker Compose utilisée ne prenait pas en charge de manière fluide Symfony 3.1, limitant ainsi la portabilité et la facilité de déploiement de l'application. En optant pour une refonte totale en Symfony 6.4, nous avons pu contourner ces problèmes de compatibilité et bénéficier d'une intégration plus harmonieuse avec notre infrastructure Docker, facilitant ainsi le déploiement et la gestion de l'application dans des environnements de production.

## 4. Évaluation de la qualité du code :

### Analyse avec Codacy :

- Note obtenue : A
- Pourcentage d'issues mineures : 3%
- Nombre total d'issues détectées : 52 (Code style : 47 issues, Unused code : 5 issues)

P8-master C main ▼

Issues 🔍 ●  
29 %

**285** total issues



[See all issues](#)

OCphp\_P8\_ToDoAndCo A main ▼

Issues 🔍 ●  
3 % ▼ -19 %

**52** total issues


















Comparaison avec l'ancienne version de l'application : Avant la refonte de l'application, l'évaluation de la qualité du code avec Codacy donnait une note C, avec un pourcentage beaucoup plus élevé d'issues (29%). Sur un total de 285 issues détectées, la grande majorité (252) étaient liées au code style, tandis que quelques-unes étaient liées à la sécurité et à l'unused code.

### Tests automatisés :

		Lines
Total	<div></div>	96.53%
■ Controller	<div></div>	94.44%
■ Entity	<div></div>	100.00%
■ Form	<div></div>	100.00%
■ Repository	<div></div>	100.00%

- Implémentation de tests unitaires et fonctionnels avec PHPUnit
- Couverture de code : 96.53%

		Code Coverage					
		Lines		Functions and Methods		Classes and Traits	
Total		96.53%	167 / 173		95.00%	38 / 40	 80.00% 8 / 10
■ Controller		94.44%	102 / 108		81.82%	9 / 11	 50.00% 2 / 4
■ Entity		100.00%	35 / 35		100.00%	24 / 24	 100.00% 2 / 2
■ Form		100.00%	23 / 23		100.00%	2 / 2	 100.00% 2 / 2
■ Repository		100.00%	7 / 7		100.00%	3 / 3	 100.00% 2 / 2

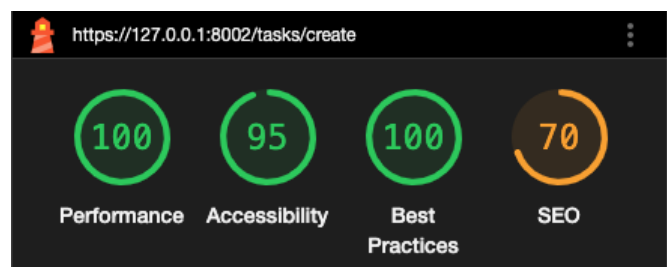
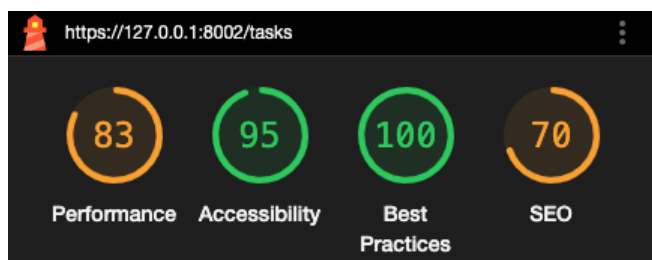
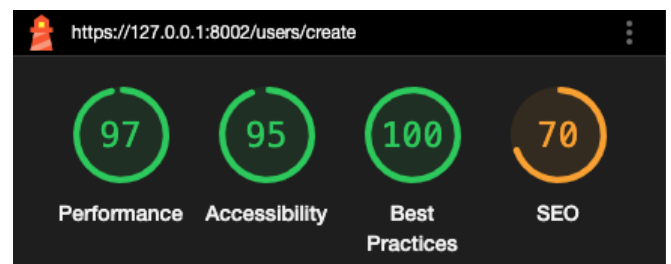
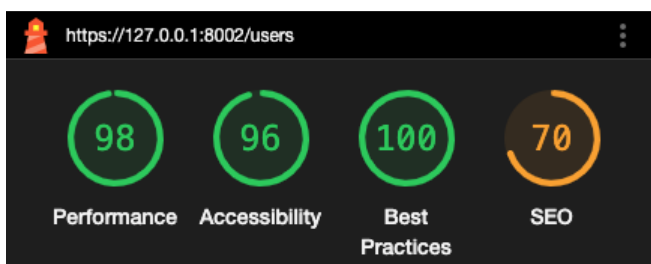
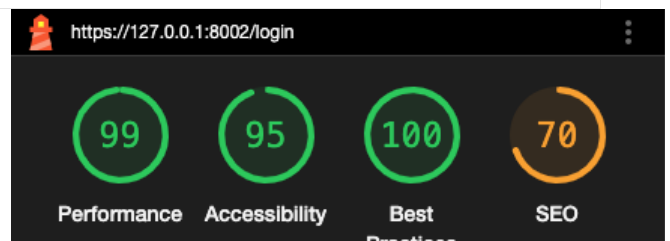
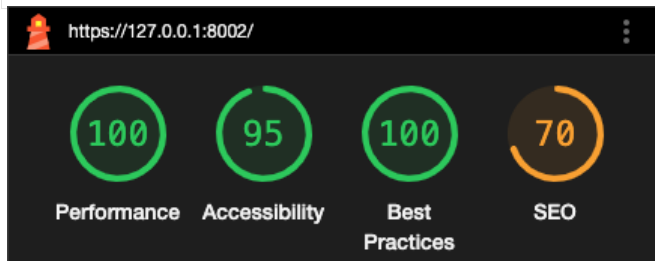
**Conformité aux bonnes pratiques de développement Symfony :** Lors de la refonte de l'application, nous avons veillé à respecter les bonnes pratiques de développement Symfony, garantissant ainsi la maintenabilité, la lisibilité et la robustesse du code. Voici quelques exemples de bonnes pratiques que nous avons suivies :

- **Noms de variables de fonctions et de classes significatifs :** Les noms des variables, des fonctions et des classes sont choisis de manière à être descriptifs et significatifs, facilitant ainsi la compréhension du code par les autres développeurs.
- **Convention de nommage (camelCase et snake\_case) :** Nous avons suivi la convention de nommage recommandée par Symfony, utilisant le camelCase pour les noms de variables et de fonctions, et le snake\_case pour les noms de tables et de colonnes de base de données.
- **Indentation claire et mise en forme :** Le code est indenté de manière cohérente et la mise en forme est soignée, ce qui facilite la lecture et la compréhension du code.
- **Commentaires :** Des commentaires sont ajoutés lorsque nécessaire pour expliquer le fonctionnement complexe ou non intuitif du code, ainsi que pour documenter les décisions de conception importantes.
- **Single Responsibility Principle (SRP) :** Nous avons appliqué le principe de responsabilité unique, en veillant à ce que chaque classe et chaque méthode aient une seule responsabilité, ce qui facilite la maintenance et l'évolutivité du code.
- **DRY (Don't Repeat Yourself) :** Nous avons évité la duplication de code en extrayant les fonctionnalités répétitives dans des fonctions ou des classes réutilisables, favorisant ainsi la cohérence et la facilité de maintenance du code.
- **Gestion des erreurs et des exceptions :** Nous avons mis en place une gestion appropriée des erreurs et des exceptions, en capturant et en traitant les erreurs de manière appropriée pour assurer la robustesse de l'application.
- **Documentation :** Une documentation claire et concise a été fournie, expliquant le fonctionnement général de l'application, les choix de conception et les instructions pour l'installation et la configuration.
- **Sécurité (csrf token) :** Nous avons intégré des mesures de sécurité telles que la protection contre les attaques CSRF (Cross-Site Request Forgery) en utilisant les fonctionnalités de sécurité fournies par Symfony.

La refonte de l'application a permis de réaliser des améliorations significatives en termes de qualité du code. La note obtenue avec Codacy est passée de C à A, et le pourcentage d'issues mineures a été réduit à seulement 3%. Les principales sources d'issues détectées concernent le code style, avec 47 issues, et l'unused code, avec 5 issues. Il est important de noter que ces issues de unused code correspondent à des déclarations de variables nécessaires pour Symfony, malgré l'analyse de Codacy. Les tests automatisés ont été implémentés avec succès, Une bonne couverture de code, telle que celle que nous avons atteinte avec une couverture de 96.53%, garantit une meilleure fiabilité et maintenabilité de l'application, en identifiant les zones de code non testées et en réduisant les risques d'erreurs et de régressions. De plus, le code est conforme aux bonnes pratiques de développement Symfony, garantissant une maintenabilité et une évolutivité optimales de l'application. Ces résultats témoignent des efforts déployés pour améliorer la qualité, la robustesse et la maintenabilité du code de l'application.

## 5. Évaluation de la performance de l'application

**Analyse avec Lighthouse : résultats et recommandations** L'outil Lighthouse a été utilisé pour évaluer la performance de l'application sous différents aspects, tels que la performance web, l'accessibilité, les bonnes pratiques SEO, etc. Les résultats de cette analyse fournissent une vue d'ensemble des performances actuelles de l'application et des recommandations pour les améliorer.



▲ Enable text compression — Potential savings of 325 KiB

Text-based resources should be served with compression (gzip, deflate or brotli) to minimize total network bytes. [Learn more about text compression.](#) [FCP] [LCP]

URL	Transfer Size	Potential Savings
0.1 [1st Party]	395.7 KiB	325.1 KiB
/css/bootstrap.min.css (127.0.0.1)	227.5 KiB	197.1 KiB
/js/bootstrap.bundle.min.js (127.0.0.1)	78.8 KiB	55.6 KiB
/tasks (127.0.0.1)	62.1 KiB	50.2 KiB
/_wdt/9db6d2 (127.0.0.1)	27.3 KiB	22.3 KiB

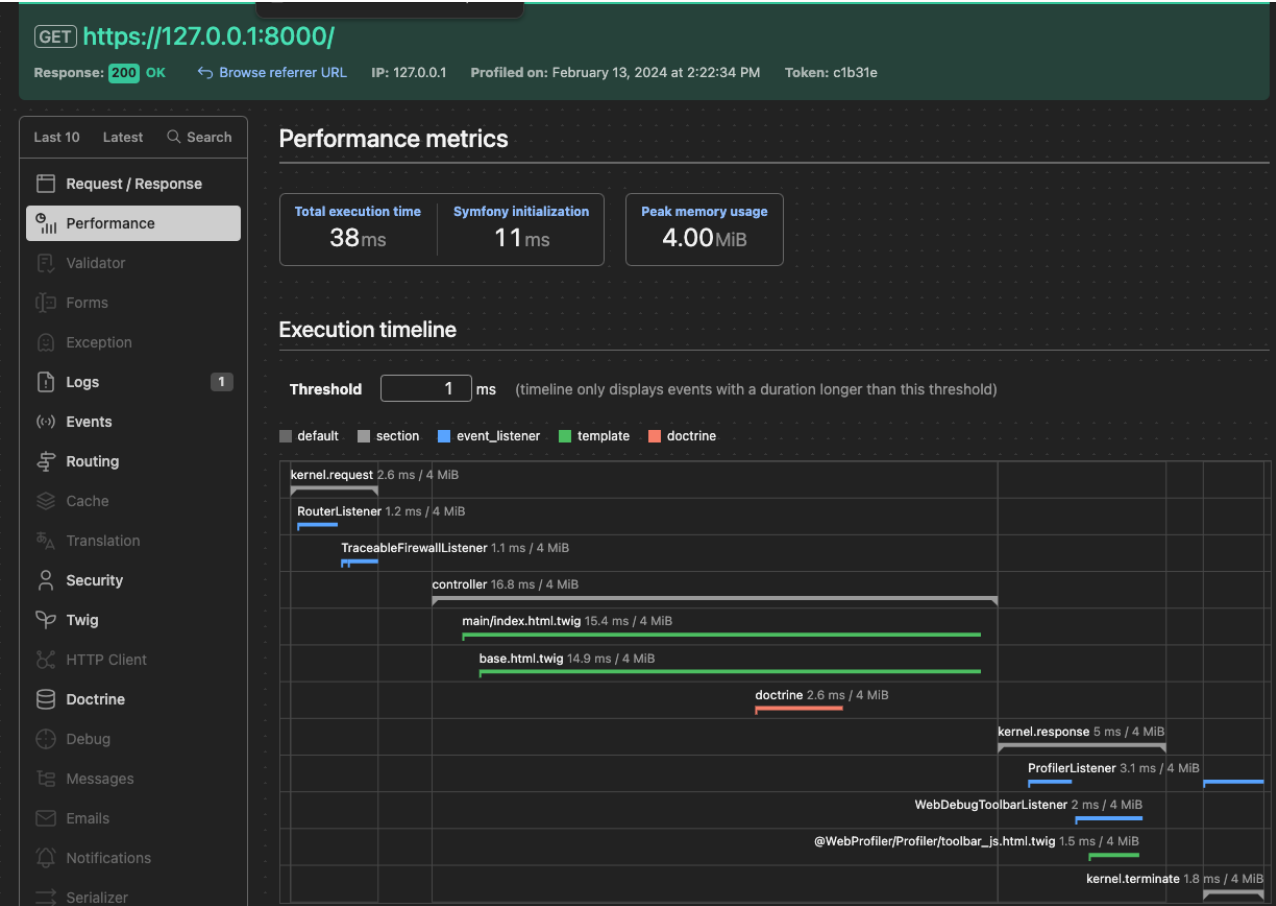
▲ Serve images in next-gen formats — Potential savings of 29 KiB

Image formats like WebP and AVIF often provide better compression than PNG or JPEG, which means faster downloads and less data consumption. [Learn more about modern image formats.](#)

■ Minify CSS — Potential savings of 3 KiB	▼
■ Minify JavaScript — Potential savings of 124 KiB	▼
■ Serve static assets with an efficient cache policy — 5 resources found	▼
■ Image elements do not have explicit width and height	▼
■ Eliminate render-blocking resources — Potential savings of 0 ms	▼
■ Reduce unused CSS — Potential savings of 216 KiB	▼
■ Reduce unused JavaScript — Potential savings of 88 KiB	▼
■ Efficiently encode images — Potential savings of 5 KiB	▼
■ Avoid serving legacy JavaScript to modern browsers — Potential savings of 8 KiB	▼



**Utilisation du Symfony Profiler : optimisations possibles** Le Symfony Profiler a également été utilisé pour identifier les zones de l'application nécessitant des optimisations. Les données fournies par le profiler, telles que les requêtes SQL, le temps de chargement des pages et l'utilisation des ressources système, ont permis d'identifier les goulets d'étranglement et les points critiques de l'application en termes de performance (voir le dossier Symfony Profiler annexe pour l'intégralité des captures).



## Conclusion

L'audit de qualité et de performance réalisé sur l'application ToDo & Co a permis de mettre en lumière plusieurs aspects positifs. Tout d'abord, il est important de souligner la bonne qualité du code ainsi que les performances optimales de l'application, telles que révélées par les analyses approfondies menées avec Codacy, Lighthouse et le Symfony Profiler. La décision de procéder à une refonte totale de l'application Symfony 3.1 vers Symfony 6.4 s'est avérée être une initiative judicieuse, comme en témoignent les résultats obtenus. La migration vers une version plus récente a permis de résoudre de nombreux problèmes hérités de la version précédente, mettant ainsi la dette technologique à zéro et assurant une base solide pour le développement futur de l'application.

Cependant, pour garantir que l'application continue de répondre aux exigences élevées en matière de qualité et de performance, il est essentiel d'établir un processus de suivi et d'évaluation des progrès. À cette fin, je recommande la mise en place d'une méthode de suivi continue, impliquant des revues régulières du code, des tests de performance périodiques et des analyses approfondies de l'expérience utilisateur. Cette approche permettra de détecter rapidement tout problème émergent et d'apporter des correctifs appropriés, assurant ainsi que l'application reste à la pointe de la technologie.

En outre, des perspectives d'amélioration pour l'application peuvent être explorées pour répondre aux besoins évolutifs des utilisateurs. Cela pourrait inclure l'ajout de nouvelles fonctionnalités en réponse aux retours des utilisateurs, l'optimisation de l'interface utilisateur pour une meilleure expérience utilisateur, ou encore l'intégration de technologies émergentes pour rester à la pointe de l'innovation. L'utilisation d'outils comme l'extension Green It pour Chrome permettrait d'améliorer également l'application sur le plan du numérique responsable.

En conclusion, l'audit a confirmé la robustesse et la fiabilité de l'application ToDo & Co, tout en identifiant des pistes d'amélioration pour assurer sa pérennité à long terme. En adoptant une approche proactive de suivi et d'évaluation, et en restant ouverts aux opportunités d'amélioration continue, nous sommes bien positionnés pour répondre aux défis futurs et offrir une expérience exceptionnelle à nos utilisateurs.