2024-2025

TASKSTEP : AUDIT D'OPTIMISATION DE PERFORMANCES ET D'ENVIRONNEMENT

TRISTAN DAL-MOLIN

- Tristan DAL MOLIN
- Moulay-Wassim ALAOUI
- Jules DUTRION
- Matteo DE MARCO
- Wassim DIOURI

BUT2 Informatique

IUT DIJON AUXERRE NEVERS
UNIVERSITÉ DE BOURGOGNE
Révélateur de talents depuis 1968

UB
UNIVERSITÉ DE BOURGOGNE

I. Introduction

Ce rapport présente l'audit d'optimisation des performances et de l'impact environnemental de l'application TaskStep. Il repose sur l'analyse de deux fonctionnalités clés, testées dans des contextes d'utilisation variés (application légère et lourde). L'audit s'appuie sur des référentiels reconnus, notamment GreenIT pour la partie environnementale, et Lighthouse pour les mesures de performance. Une comparaison entre l'application dans son état originel et après optimisation sera effectuée durant cet audit afin de déterminer à quel point nous avons pu faire gagner cette application en performances.

II. Évaluation de l'empreinte environnementale

Pour l'analyse qui suit, nous nous appuyons sur le référentiel GreenIT, utilisé comme checklist d'évaluation. Ce référentiel constitue également la base de mesure de l'empreinte environnementale de l'application web TaskStep.

A. Contexte

Afin d'évaluer l'empreinte environnementale de l'application web TaskStep, nous avons choisi d'analyser deux parcours utilisateurs correspondant à des fonctionnalités essentielles de l'application. Ce choix permet de représenter des cas d'usage concrets et fréquents, vécus au quotidien par les utilisateurs.

Pour garantir la pertinence des résultats, chaque scénario a été testé dans deux contextes distincts :

- Une application légère, avec très peu de tâches en base (application "quasi vierge")
- Une application conséquente, contenant un volume important de données (~3000 à 3600 tâches)

Cette approche comparative permet de juger si l'application reste performante et éco-efficiente en fonction de sa charge réelle. Elle reflète à la fois des usages standards et des cas plus extrêmes, afin de garantir un audit complet et réaliste.

B. Présentation des critères d'audit

Pour réaliser cet audit, nous nous sommes appuyés sur les critères proposés par l'extension GreenIT Analysis, outil de référence dans l'évaluation de la performance environnementale des applications web.

Cette extension se base sur un ensemble de bonnes pratiques techniques, regroupées dans un référentiel d'éco-conception. Ces pratiques couvrent différents aspects de l'optimisation numérique tels que :

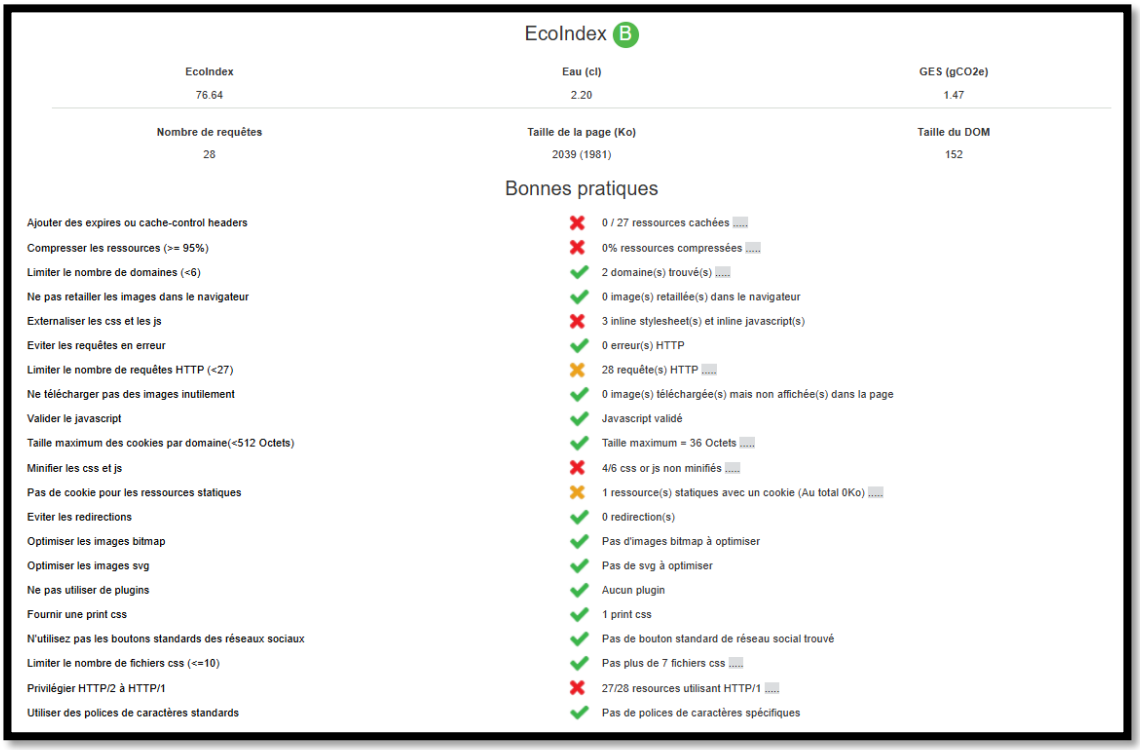
- La réduction du poids des ressources (compression, minification)
- Le contrôle des requêtes HTTP (quantité, redirections, domaines)
- L'optimisation des images (format, taille, absence de redimensionnement navigateur)
- L'usage de techniques de mise en cache, de headers adaptés, ou encore de ressources externes bien gérées
- L'application de règles d'accessibilité mobile (ex. : meta viewport)

Ces critères ont servi de base pour identifier les forces de l'application (c'est-à-dire les bonnes pratiques déjà en place) ainsi que les axes d'amélioration (vulnérabilités environnementales).

C. Fonctionnalité : Ajouter une tâche

Scénario d'utilisation : Un étudiant décide d'ajouter une tâche « Rendre le rapport » avec une date limite fixée au 20 Avril.

Résultat d'analyse de la fonctionnalité :



EcoIndex B

EcoIndex	Eau (cl)	GES (gCO2e)
76.64	2.20	1.47

Nombre de requêtes	Taille de la page (Ko)	Taille du DOM
28	2039 (1981)	152

Bonnes pratiques

Ajouter des expires ou cache-control headers	✗ 0 / 27 ressources cachées
Compresser les ressources (>= 95%)	✗ 0% ressources compressées
Limiter le nombre de domaines (<6)	✓ 2 domaine(s) trouvé(s)
Ne pas retailler les images dans le navigateur	✓ 0 image(s) retaillée(s) dans le navigateur
Externaliser les css et les js	✗ 3 inline stylesheet(s) et inline javascript(s)
Eviter les requêtes en erreur	✓ 0 erreur(s) HTTP
Limiter le nombre de requêtes HTTP (<27)	✗ 28 requête(s) HTTP
Ne télécharger pas des images inutilement	✓ 0 image(s) téléchargée(s) mais non affichée(s) dans la page
Valider le javascript	✓ Javascript validé
Taille maximum des cookies par domaine(<512 Octets)	✓ Taille maximum = 36 Octets
Minifier les css et js	✗ 4/6 css or js non minifiés
Pas de cookie pour les ressources statiques	✗ 1 ressource(s) statiques avec un cookie (Au total 0Ko)
Eviter les redirections	✓ 0 redirection(s)
Optimiser les images bitmap	✓ Pas d'images bitmap à optimiser
Optimiser les images svg	✓ Pas de svg à optimiser
Ne pas utiliser de plugins	✓ Aucun plugin
Fournir une print css	✓ 1 print css
N'utilisez pas les boutons standards des réseaux sociaux	✓ Pas de bouton standard de réseau social trouvé
Limiter le nombre de fichiers css (<=10)	✓ Pas plus de 7 fichiers css
Privilégier HTTP/2 à HTTP/1	✗ 27/28 resources utilisant HTTP/1
Utiliser des polices de caractères standards	✓ Pas de polices de caractères spécifiques

Figure 1 : Note EcoIndex Ajout d'une tâche

Dans le cadre de ce scénario d'utilisation, l'analyse de l'outil GreenIT a révélé plusieurs bonnes pratiques d'éco-conception déjà respectées :

- Aucune image redimensionnée côté navigateur
- Aucune erreur HTTP détectée
- Aucune image inutilement téléchargée
- JavaScript correctement validé
- Taille des cookies respectée (36 octets, bien en dessous du seuil de 512)
- Aucune redirection détectée
- Aucune image bitmap à optimiser
- Aucune image SVG à optimiser
- Aucun plugin tiers utilisé
- Feuille de style dédiée à l'impression présente (print.css)
- Aucun bouton standard de réseau social détecté
- Nombre de fichiers CSS limité (moins de 10)
- Polices de caractères standards utilisées
- Très bonne note ExoIndex : B

Ces bonnes pratiques montrent que, dans un contexte simple, l'application est globalement sobre et bien pensée d'un point de vue environnemental.

L'analyse a également permis de relever des vulnérabilités :

Vulnérabilité observée	Degré de risque	Solution proposée
Absence de cache HTTP (0/27 ressources cachées)	Élevé	Ajouter des expires ou headers cache-control
Aucune ressource compressée	Élevé	Compresser les ressources
Ressources non externalisées (3 styles inline)	Moyen	Externaliser les CSS et JS dans des fichiers dédiés
CSS et JS non minifiés (4/6)	Moyen	Minifier les fichiers CSS et JS
Utilisation partielle de HTTP/2 (27/28)	Moyen	Privilégier HTTP/2 pour toutes les ressources
Nombre de requêtes élevé (28)	Faible	Réduire le nombre de requêtes HTTP (<27)
Présence d'un cookie sur une ressource statique	Faible	Éviter les cookies pour les ressources statiques

Ces vulnérabilités, bien que pour certaines d'entre elles mineures, peuvent avoir un impact significatif sur la performance environnementale de l'application à grande échelle. Les plus critiques concernent l'absence de mise en cache et l'absence de compression des ressources, qui augmentent inutilement les échanges réseaux à chaque chargement. D'autres points comme la non minification des fichiers ou l'usage partiel de HTTP/2 sont relativement simples à corriger et permettraient de réduire la consommation de bande passante et d'énergie.

Scénario d'utilisation : Un étudiant décide d'ajouter une tâche « Rendre le rapport » avec une date limite fixée au 20 Avril parmi un nombre conséquent de tâches.

Résultat d'analyse de la fonctionnalité :

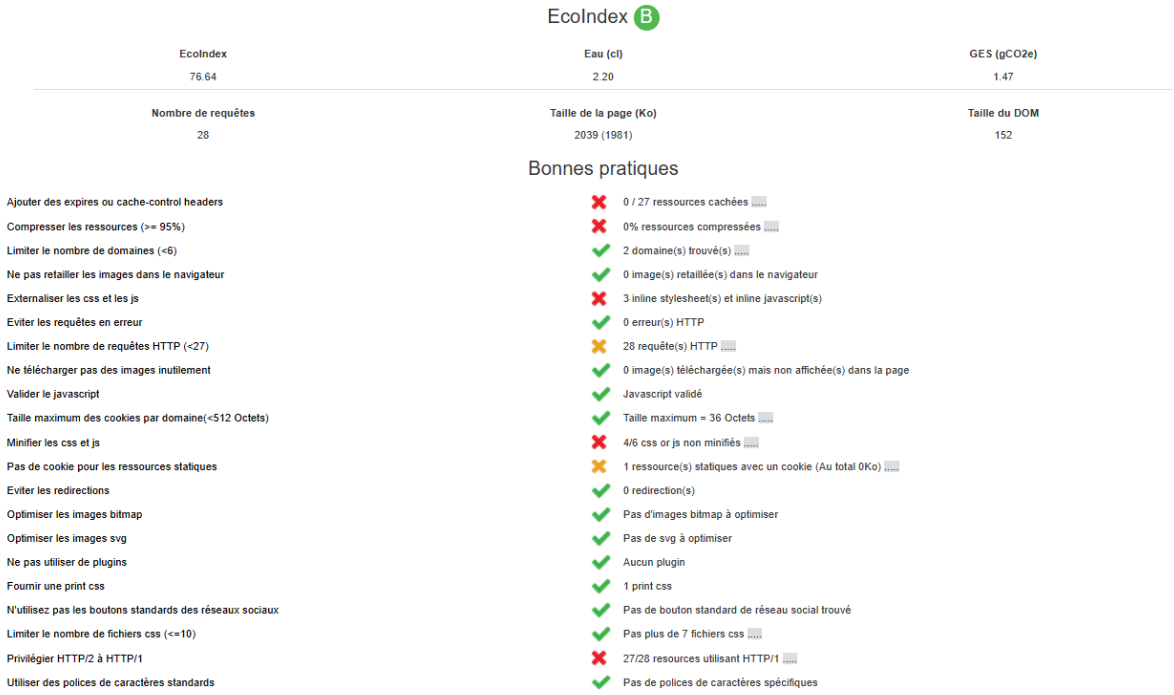


Figure 2 : Note EcoIndex ajout d'une tâche parmi de nombreuses tâches

L'audit révèle que les résultats en matière d'éco-conception sont strictement identiques entre l'ajout d'une tâche dans un environnement léger et dans un environnement lourd (plusieurs milliers de tâches en base).

Cela s'explique par le fait que la page concernée (edit.php) reste techniquement inchangée quelle que soit la volumétrie des données. Elle se contente d'afficher un formulaire simple, sans interagir avec l'ensemble des éléments stockés. Ainsi :

- Le DOM généré est identique
- Les ressources chargées sont les mêmes
- Le nombre de requêtes et les scripts exécutés ne varient pas

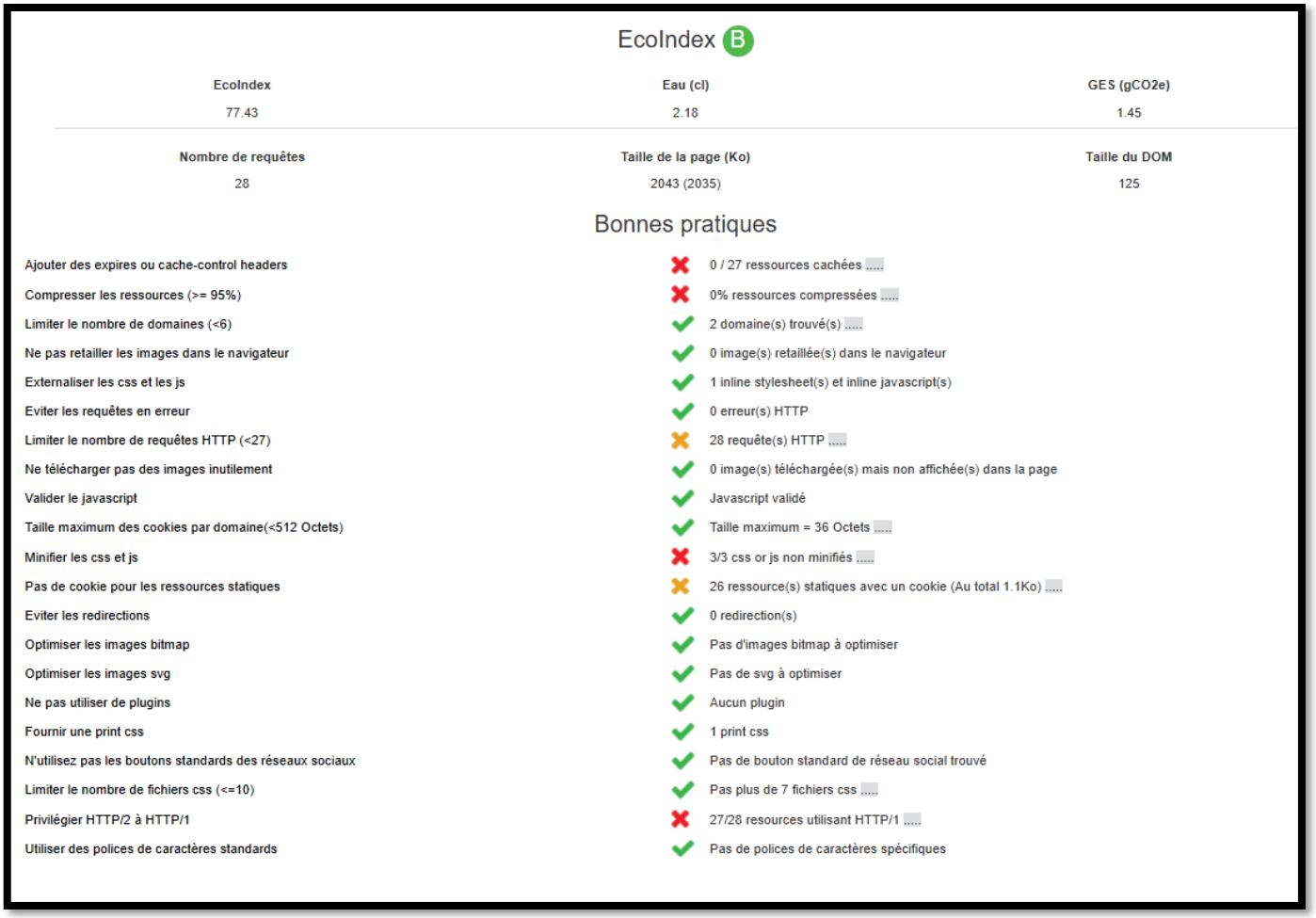
Cette stabilité est un bon indicateur d'éco-conception : l'application ne surcharge pas inutilement ses pages en fonction du contexte ou de la charge serveur pour cette fonctionnalité.

Toutefois, il est important de noter que les vulnérabilités détectées restent présentes dans les deux cas, ce qui indique qu'elles sont liées à la structure globale de l'application, et non à un état spécifique de la base de données.

D. Fonctionnalité : Visualisation des tâches

Scénario d'utilisation : Un étudiant veut consulter les tâches qu'il a dans la rubrique « this week » représentant ses tâches pour le week-end avec très peu de tâches présentes.

Résultat d'analyse de la fonctionnalité :



EcoIndex B

EcoIndex	Eau (cl)	GES (gCO2e)
77.43	2.18	1.45

Nombre de requêtes	Taille de la page (Ko)	Taille du DOM
28	2043 (2035)	125

Bonnes pratiques

Ajouter des expires ou cache-control headers	✗ 0 / 27 ressources cachées ....
Compresser les ressources (>= 95%)	✗ 0% ressources compressées ....
Limiter le nombre de domaines (<6)	✓ 2 domaine(s) trouvé(s) ....
Ne pas retailler les images dans le navigateur	✓ 0 image(s) retaillée(s) dans le navigateur
Externaliser les css et les js	✓ 1 inline stylesheet(s) et inline javascript(s)
Eviter les requêtes en erreur	✓ 0 erreur(s) HTTP
Limiter le nombre de requêtes HTTP (<27)	✗ 28 requête(s) HTTP ....
Ne télécharger pas des images inutilement	✓ 0 image(s) téléchargée(s) mais non affichée(s) dans la page
Valider le javascript	✓ Javascript validé
Taille maximum des cookies par domaine(<512 Octets)	✓ Taille maximum = 36 Octets ....
Minifier les css et js	✗ 3/3 css or js non minifiés ....
Pas de cookie pour les ressources statiques	✗ 26 ressource(s) statiques avec un cookie (Au total 1.1Ko) ....
Eviter les redirections	✓ 0 redirection(s)
Optimiser les images bitmap	✓ Pas d'images bitmap à optimiser
Optimiser les images svg	✓ Pas de svg à optimiser
Ne pas utiliser de plugins	✓ Aucun plugin
Fournir une print css	✓ 1 print css
N'utilisez pas les boutons standards des réseaux sociaux	✓ Pas de bouton standard de réseau social trouvé
Limiter le nombre de fichiers css (<=10)	✓ Pas plus de 7 fichiers css ....
Privilégier HTTP/2 à HTTP/1	✗ 27/28 resources utilisant HTTP/1 ....
Utiliser des polices de caractères standards	✓ Pas de polices de caractères spécifiques

Figure 3 : Note EcoIndex de la visualisation des tâches

De la même façon que pour la tâche "Ajout d'une tâche", l'analyse de l'outil GreenIT a permis de relever, pour le scénario de visualisation des tâches dans un environnement léger, plusieurs bonnes pratiques d'éco-conception respectées :

- Limitation du nombre de domaines (2 domaines trouvés)
- Aucune image redimensionnée dans le navigateur
- Aucune erreur HTTP détectée
- Aucune image inutilement téléchargée
- JavaScript correctement validé
- Taille des cookies respectée (36 octets)
- Aucune redirection détectée
- Aucune image bitmap à optimiser
- Aucune image SVG à optimiser
- Aucun plugin tiers utilisé
- Présence d'une feuille de style dédiée à l'impression (print.css)
- Aucun bouton standard de réseau social détecté
- Moins de 10 fichiers CSS utilisés
- Polices de caractères standards employées

Cela montre que dans un environnement minimal, l'application respecte plusieurs principes essentiels de sobriété numérique.

Cependant notre analyse a quand même permis de relever des vulnérabilités :

Vulnérabilité observée	Degré de risque	Solution proposée
Aucune ressource mise en cache (0/27 ressources)	Élevé	Ajouter des headers cache-control ou expires
Aucune ressource compressée	Élevé	Activer la compression des ressources
Présence de 3 fichiers CSS/JS inline non externalisés	Moyen	Externaliser les fichiers dans des ressources dédiées
Nombre de requêtes élevé (28 requêtes)	Faible	Réduire ou regrouper les requêtes pour rester sous le seuil de 27
Fichiers CSS/JS non minifiés (3/3 non minifiés)	Moyen	Minifier les fichiers CSS et JavaScript
Cookie présent sur une ressource statique	Faible	Éviter d'envoyer des cookies sur les fichiers statiques
Utilisation partielle de HTTP/2 (27/28 ressources en HTTP/1)	Moyen	Passer l'ensemble des ressources au protocole HTTP/2

Ces vulnérabilités, bien qu'en partie mineures, révèlent des axes d'amélioration concrets pour optimiser l'impact environnemental de l'application. Leur correction permettrait de réduire significativement la consommation de ressources réseau sans altérer l'expérience utilisateur

Scénario d'utilisation : Un étudiant veut consulter les tâches qu'il a dans la rubrique « this week » représentant ses tâches pour le week-end avec déjà un nombre important de tâches présentes.

Résultat d'analyse de la fonctionnalité :

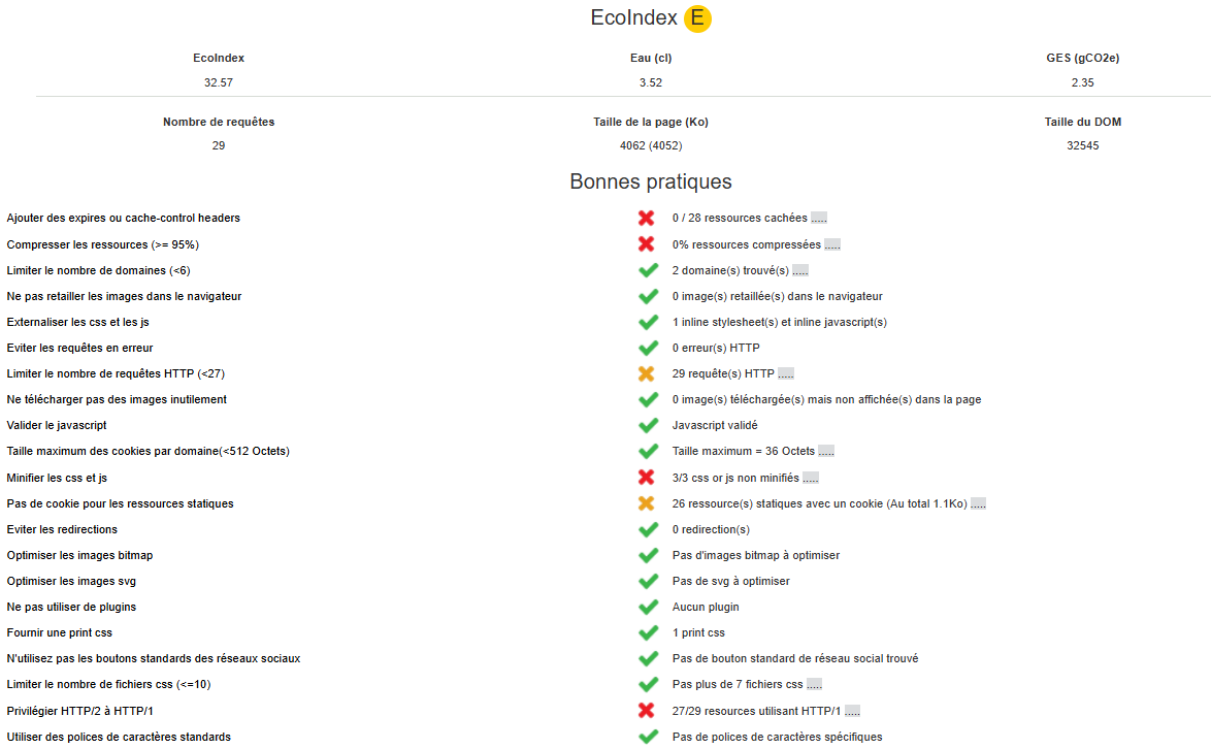


Figure 4 : NoteEcoIndex visualisation de nombreuses tâches

Bien que les bonnes pratiques techniques soient globalement respectées et identiques à celles observées dans un environnement léger, les indicateurs environnementaux se dégradent fortement lorsque la base de données contient un grand nombre de tâches (Dans notre cadre on a testé avec ~3600 tâches).

On constate notamment :

- Une taille de page extrêmement élevée (plus de 4 Mo, soit le double de quand on était en environnement léger), ce qui est problématique pour une simple liste textuelle.
- Un DOM surchargé avec plus de 32 000 éléments, ce qui entraîne une consommation excessive de ressources pour l'affichage.
- Un EcoIndex qui chute à E, traduisant une expérience particulièrement énergivore.

Ces résultats indiquent que l'architecture de l'application n'est pas adaptée à une montée en charge. L'absence de pagination, de chargement différé ou de limitation de contenu affiché provoque une explosion des ressources utilisées côté client, ce qui est contraire aux principes de sobriété numérique.

Des optimisations sont indispensables pour garantir la soutenabilité de l'application à long terme.

E. Conclusions d'audit

L'audit environnemental de l'application TaskStep met en évidence une base technique globalement stable et conforme à plusieurs bonnes pratiques d'éco-conception. Ces points positifs sont observés de manière constante, quel que soit le scénario ou la charge de données.

Cependant, des vulnérabilités récurrentes sont identifiées (absence de cache, compression désactivée, fichiers non minifiés, HTTP/2 partiel) et, surtout, une forte dégradation des indicateurs apparaît en environnement chargé : poids de la page, complexité du DOM, et chute de l'EcoIndex.

Cela souligne une absence d'adaptation à la montée en charge, en contradiction avec les principes de sobriété numérique. Des optimisations techniques ciblées sont nécessaires pour limiter l'impact environnemental de l'application à long terme.

F. Optimisation de l'application

Suite à l'audit initial, plusieurs optimisations ont été mises en place afin de pouvoir corriger les vulnérabilités identifiées et ainsi améliorer à la fois l'impact environnemental et les performances de l'application.

Afin de réaliser une comparaison très cohérente, après avoir réalisé tout le développement du site nous avons à nouveau réaliser des analyses avec l'outil GreenIT des 2 scénarios d'utilisation les plus fréquents pour l'application c'est-à-dire visualiser une tâche ainsi qu'ajouter une tâche (dans un premier temps dans un environnement léger ~3/5 tâches déjà présentes ainsi que dans un environnement lourd ~3000 tâches) afin de pouvoir juger si l'application est adaptée pour une monté de données.

En ce qui concerne l'ajout d'une tâche que ce soit dans un environnement léger ou lourd l'application de base Taskstep présentait déjà des données satisfaisantes (score EcoIndex B, 76.64 note EcoIndex) cependant le nombre de requêtes restaient très élevées par rapport à la tâche demandée. Notre implémentation a donc pu permettre d'améliorer l'impact environnemental de l'application :

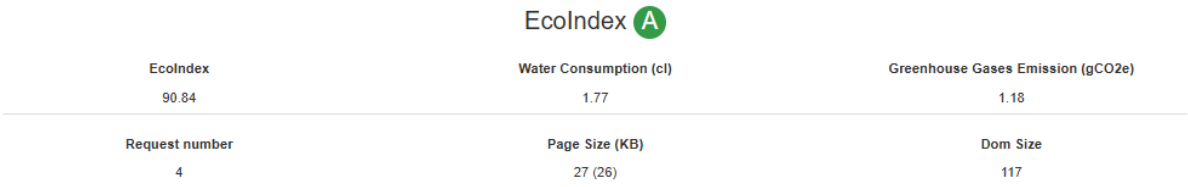


Figure 2 : Ajout d'une tâche sur un environnement léger

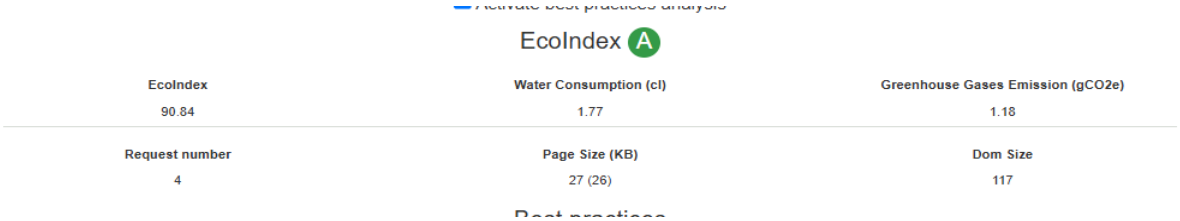


Figure 5 : Ajout d'une tâche au sein d'un environnement lourd

En ce qui concerne la seconde tâche, c'est-à-dire visualiser des tâches que ce soit dans un environnement léger ou lourd, l'application avait un score satisfait dans un environnement léger 77.43 et un score B EcoIndex . Cependant on a pu remarquer l'application n'était pas adapté pour une montée en charge car sous un environnement lourd le score EcoIndex était à 32.57 et E montrant ainsi un impact environnemental de l'application très mauvais sur un environnement lourd. À la suite de notre implémentation nous avons nettement pu améliorer ces résultats :

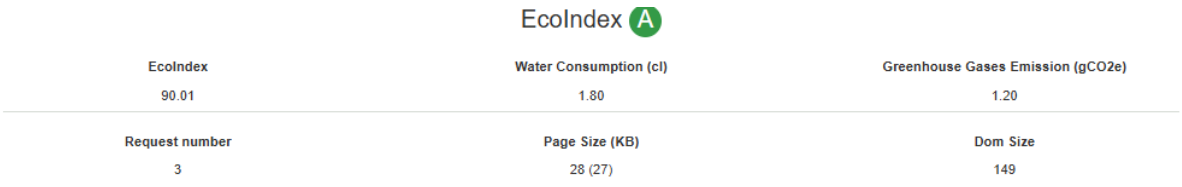


Figure 6 : Visualisation d'une tâche dans un environnement léger

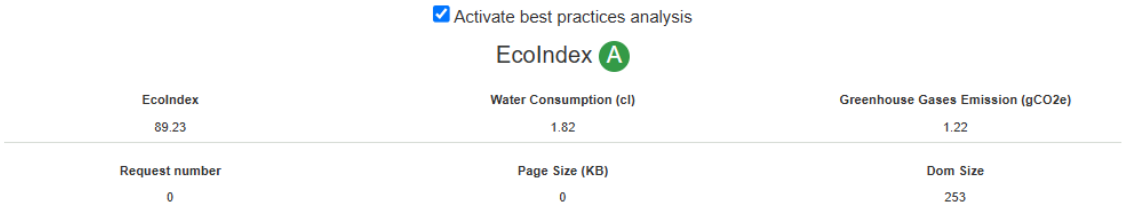


Figure 7 : Visualisation d'une tâche dans un environnement lourd

G. Conclusion comparative

L'évolution des scores EcoIndex obtenus après optimisation témoigne d'un changement structurel profond dans le fonctionnement de l'application. Là où TaskStep se contentait auparavant d'être « sobre » en environnement léger, elle est désormais performante et éco-responsable quel que soit le volume de données.

Avant les améliorations, l'application atteignait un niveau critique dès qu'elle devait gérer une grande quantité de tâches. L'affichage sans limitation provoquait une explosion du DOM, un poids de page supérieur à 4 Mo, et un EcoIndex de niveau E, indiquant une consommation de ressources excessive — donc peu soutenable écologiquement.

La mise en œuvre de techniques simples mais efficaces (pagination, regroupement des requêtes, réduction du contenu chargé) a permis de stabiliser l'affichage et de maîtriser les ressources consommées, même en environnement très chargé. Le passage à un EcoIndex A dans tous les scénarios testés, y compris les plus lourds, ne signifie pas seulement un bon score il permet de démontrer que la charge affichée est désormais proportionnée à l'usage réel et que l'application est adaptée à une utilisation intensive tout en restant responsable.

III. Mesures de performances

Afin de tester les performances de l'application de manière pertinente, nous avons choisi de suivre la même approche que pour l'évaluation de l'empreinte environnementale. Cette approche consiste à s'appuyer sur deux fonctionnalités essentielles et couramment utilisées :

- Ajout d'une tâche (dans une application vierge, puis dans une application contenant déjà un grand nombre de tâches)
- Visualisation des tâches (dans une base quasiment vide, puis avec une base très chargée)

Pour chaque scénario, nous avons mesuré des indicateurs de performance tels que le temps d'exécution et la vitesse de rendu, en utilisant l'outil Lighthouse. L'objectif est de déterminer si l'application reste fluide et performante en conditions réelles, aussi bien en environnement léger que sous forte charge.

Premier scénario : Un étudiant souhaite rajouter une tâche au sein de son application contenant pour l'instant aucune tâche pour se rappeler qu'il doit rendre son rapport pour ce week-end.

Indicateur utilisé	Valeur obtenue	Commentaire
Score de performance	100/100	Performance excellente
First Contentful Paint	0,3 s	Affichage initial très rapide
Largest Contentful Paint	0,4 s	Élément principal visible quasi instantanément
Total Blocking Time	0 ms	Excellent aucun blocage n'est détecté
Speed Index	0,3 s	Temps de chargement optimal

Deuxième scénario : L'étudiant souhaite faire la même chose que précédemment mais cette fois-ci il a une application avec un nombre important de tâches déjà présentes.

Indicateur utilisé	Valeur obtenue	Commentaire
Score de performance	100/100	Performance excellente
First Contentful Paint	0,3 s	Affichage initial très rapide
Largest Contentful Paint	0,4 s	Élément principal visible quasi instantanément
Total Blocking Time	0 ms	Excellent aucun blocage n'est détecté
Speed Index	0,3 s	Temps de chargement optimal

Troisième scénario : L'étudiant souhaite consulter les tâches qu'il a ajoutées au sein d'un environnement léger (qu'une seule tâche) :

Indicateur utilisé	Valeur obtenue	Commentaire
Score de performance	100/100	Performance excellente
First Contentful Paint	0,3 s	Affichage initial très rapide
Largest Contentful Paint	0,4 s	Élément principal visible quasi instantanément
Total Blocking Time	0 ms	Excellent aucun blocage n'est détecté
Speed Index	0,3 s	Temps de chargement optimal

Quatrième scénario : L'étudiant souhaite consulter ses tâches au sein d'une application avec un nombre important de tâches déjà présentes (~3600 tâches).

Indicateur utilisé	Valeur obtenue	Commentaire
Score de performance	59/100	Performance dégradée avec une forte charge
First Contentful Paint	2,0 s	Chargement initial très lent
Largest Contentful Paint	2,1 s	Élément principal visible avec un délai
Total Blocking Time	440 ms	Temps de blocage modéré, à optimiser
Speed Index	2,0 s	Rendu général ralenti

A. Conclusion sur les résultats obtenus

L'analyse des performances de l'application TaskStep, menée à travers quatre scénarios utilisateurs distincts, met en évidence un comportement très satisfaisant en environnement léger, mais une dégradation importante dès lors que l'application traite un grand volume de données.

Dans les trois premiers scénarios : ajout d'une tâche et visualisation de tâches dans une base quasi vide les résultats sont constants et excellents :

- Score de 100/100 sur Lighthouse
- Temps de chargement très courts (FCP à 0,3 s, LCP à 0,4 s)
- Aucun blocage détecté (Total Blocking Time : 0 ms)

Ces chiffres traduisent une application fluide, rapide et bien optimisée pour une utilisation simple.

En revanche, lors du scénario de visualisation avec une base très chargée (~3600 tâches), on observe une chute de performance significative :

- Score réduit à 59/100
- First Contentful Paint : 2,0 s, Largest Contentful Paint : 2,1 s
- Total Blocking Time : 440 ms

Ces dégradations sont principalement dues à un affichage en masse de toutes les tâches, sans découpage, sans limitation de rendu, et sans mécanisme d'adaptation. Cela entraîne un DOM très lourd et des sollicitations importantes du navigateur, ce qui dégrade l'expérience utilisateur.

En conclusion, l'application est bien dimensionnée pour des contextes légers, mais nécessite des ajustements techniques afin de maintenir ses performances dans des situations réelles à forte charge.

B. Recommandations techniques pour améliorer les performances

Suite aux résultats obtenus lors des différents scénarios, plusieurs pistes d'optimisation peuvent être envisagées afin de garantir la stabilité des performances de l'application, notamment en cas de forte charge.

Les recommandations principales sont les suivantes :

- Mettre en place un système de pagination : Plutôt que d'afficher toutes les tâches simultanément, il serait pertinent de charger les tâches page par page. Cela permettrait de limiter la quantité de données affichées et d'améliorer significativement les temps de chargement.
- Optimiser les requêtes vers la base de données : Actuellement, toutes les tâches sont probablement récupérées sans filtre. Il serait utile de revoir le fonctionnement des requêtes pour ne récupérer que les éléments nécessaires à l'instant donné (par exemple, les tâches de la semaine uniquement).
- Limiter le contenu affiché par défaut : Une autre solution complémentaire serait d'afficher par défaut un nombre restreint de tâches (ex. : 10 ou 20), avec la possibilité de charger la suite manuellement.
- Éviter le rechargement inutile de certaines ressources : Les scripts et fichiers CSS semblent rechargés à chaque interaction. La mise en place de mécanismes de mise en cache ou l'amélioration de la gestion des ressources côté client pourrait réduire la pression sur le navigateur.
- Prévoir une architecture plus modulable : Même si l'application reste légère dans sa logique, son comportement montre qu'elle n'est pas conçue pour monter en charge. Une réflexion globale sur l'optimisation du traitement et de l'affichage des données serait pertinente si l'application devait évoluer.

Ces recommandations visent à prévenir la dégradation des performances dans des situations plus proches d'un usage réel à long terme, et à garantir une expérience utilisateur fluide, quelle que soit la quantité de données stockées.

C. Résultats obtenus après optimisation

Suite à l'audit initial, plusieurs optimisations ont été mises en place afin de corriger les vulnérabilités identifiées et d'améliorer les performances générales de l'application, en particulier dans les cas de forte charge.

Afin de garantir une comparaison cohérente, nous avons effectué de nouvelles analyses avec l'outil Lighthouse après finalisation du développement. Les tests ont été réalisés sur les deux scénarios d'utilisation principaux : l'ajout d'une tâche et la visualisation des tâches, dans deux contextes distincts :

- Environnement léger : base contenant 3 à 5 tâches,
- Environnement lourd : base contenant environ 3000 tâches.

Les résultats ont confirmé que l'application présentait déjà des performances optimales pour la majorité des cas testés, avec des scores Lighthouse de 100/100, aussi bien pour l'ajout que pour la visualisation de tâches dans un environnement léger.

Cependant, en environnement lourd, nous avons constaté une chute notable des performances lors de la visualisation des tâches. Ce résultat était attendu : en l'absence de pagination, l'application générait une seule page contenant l'intégralité des 3000 tâches, entraînant un DOM extrêmement volumineux, une charge mémoire importante, et un ralentissement sensible du rendu. Pour remédier à cette situation, nous avons mis en place :

- Un système de pagination pour répartir l'affichage des tâches sur plusieurs pages,
- Une limitation du contenu affiché par défaut,
- Une meilleure gestion du DOM et des requêtes SQL plus ciblées.

Ces mesures ont permis à l'application de maintenir sa fluidité même sous forte charge, avec des temps de chargement réduits, une interface plus réactive, et une remontée du score Lighthouse dans les scénarios complexes :

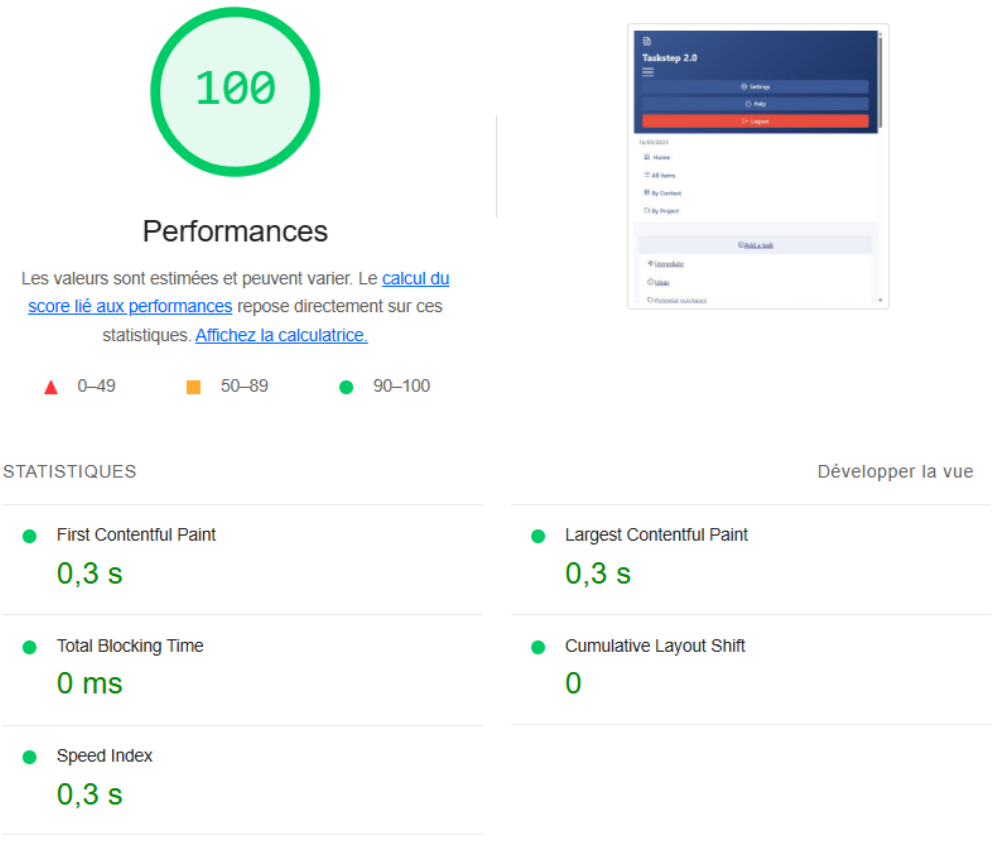


Figure 3 : Ajout d'une tâche au sein d'un environnement léger

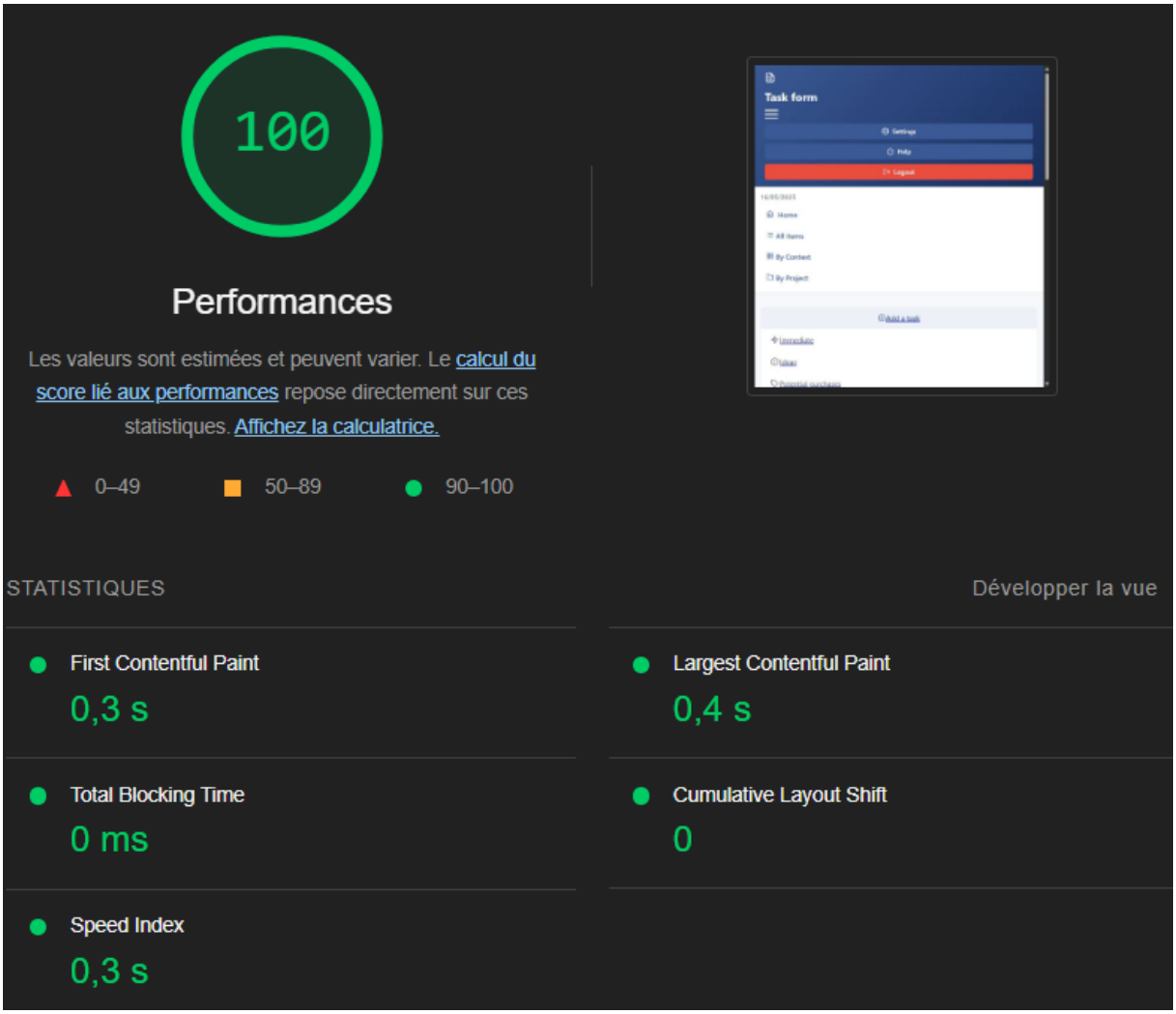


Figure 4 : Ajout d'une tâche au sein d'un environnement lourd

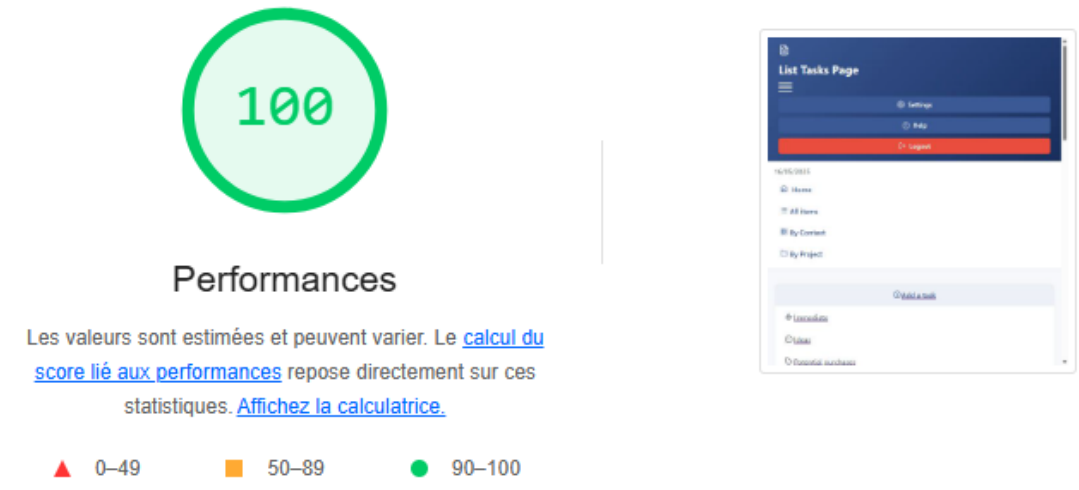


Figure 5 : Visualisation d'une tâche au sein d'un environnement léger

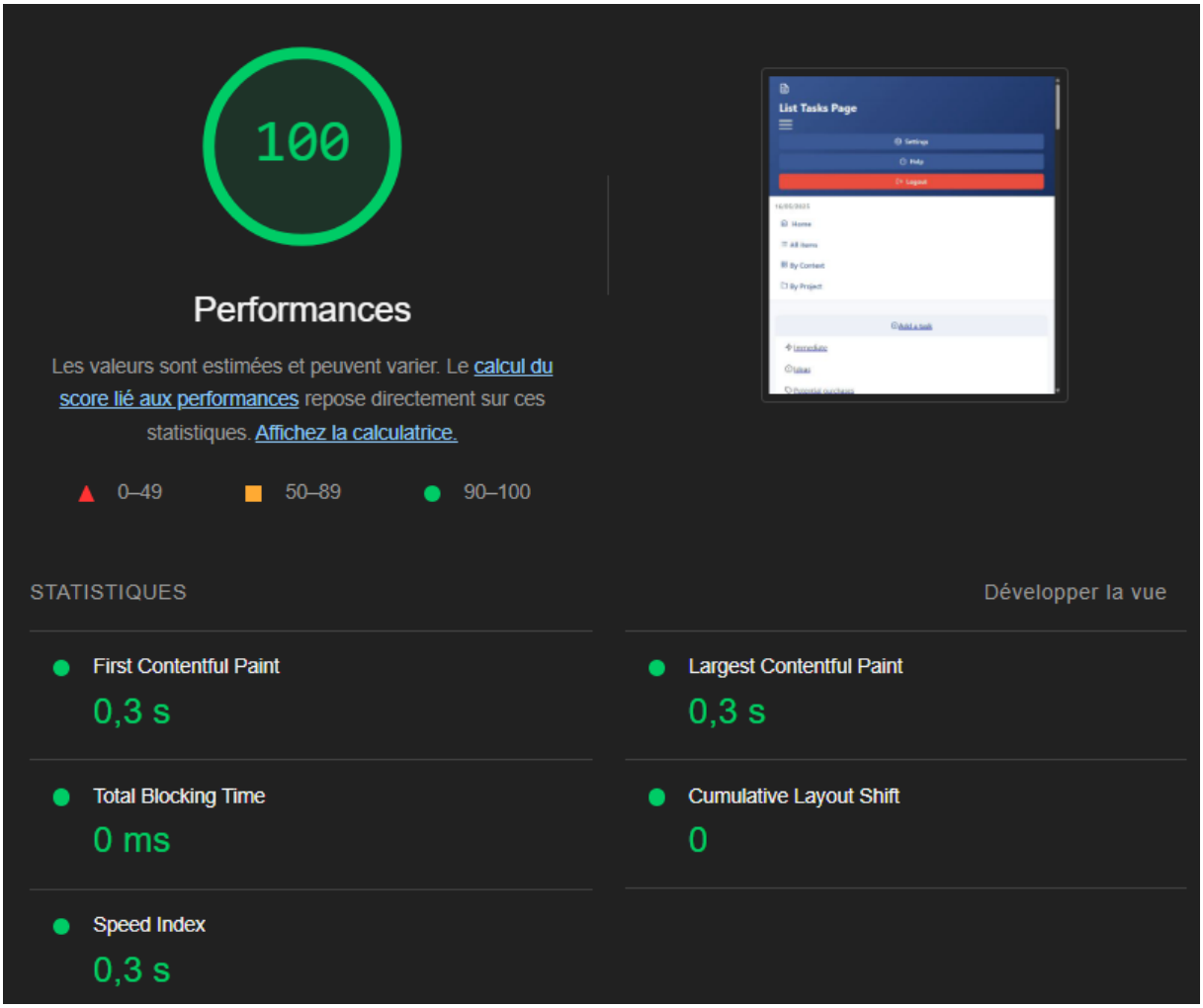


Figure 6 : Visualisation d'une tâche dans un environnement lourd

D. Conclusion comparative

L'évolution des scores Lighthouse après optimisation témoigne d'une transformation en profondeur des performances techniques de l'application TaskStep. Si elle se montrait déj à rapide et fluide en environnement léger, elle est aujourd'hui capable de maintenir un comportement optimal même sous des charges importantes.

Avant les améliorations, l'application rencontrait ses premières limites lors de la visualisation des tâches dans une base très chargée (~3000 tâches). L'absence de pagination entraînait une explosion du DOM, une charge mémoire importante, et un ralentissement notable. Le score Lighthouse chutait alors à 59/100, révélant une interface peu adaptée à un usage intensif.

Pour y remédier, nous avons mis en place des optimisations ciblées :

- Un système de pagination pour limiter le nombre de tâches affichées simultanément,
- Une réduction du contenu chargé par défaut,

- Une meilleure structuration du DOM et des requêtes SQL plus sélectives.

Ces mesures ont produit des effets immédiats : tous les scénarios, y compris les plus lourds, affichent désormais un score Lighthouse parfait de 100/100. Les captures d'écran réalisées après optimisation témoignent de temps de chargement excellents, d'un First Contentful Paint à 0,3s, et d'un Total Blocking Time nul, même avec plusieurs milliers de tâches.

Cela prouve que TaskStep est désormais parfaitement optimisée, non seulement pour une utilisation simple, mais aussi pour des contextes complexes. L'interface reste fluide, stable et réactive, quel que soit le volume de données.

IV. Tableaux des tâches

A. Environnement

Recommandations	Priorité	Complexité	Effort	Gravité
Éviter les cookies pour les ressources statiques	2	2	2	1
Privilégier HTTP/2 pour toutes les ressources	3	5	3	2
Réduire le nombre de requêtes HTTP (<27)	3	4	3	2
Externaliser les CSS et JS dans des fichiers dédiés	4	3	2	2
Minifier les fichiers CSS et JS	5	2	1	2
Ajouter des expires ou headers cache-control	6	2	1	3
Compresser les ressources	6	2	1	3

B. Performances

Recommandations	Priorité	Complexité	Effort	gravité
Activer la mise en cache des ressources statiques	3	2	1	2
Limiter le nombre d'éléments affichés par défaut	4	2	1	2
Optimiser les requêtes pour cibler les données utiles	5	4	3	3
Mettre en place un système de pagination	6	3	2	3
Adapter l'architecture pour supporter la montée en charge	6	6	4	3

V. Synthèse

Cette analyse a été réalisée dans le respect des principes d'éco-conception numérique et en s'appuyant sur des outils reconnus tels que Lighthouse et GreenIT Analysis. Les tests ont été menés sur plusieurs scénarios représentatifs des usages utilisateurs afin d'évaluer à la fois la rapidité d'exécution de l'application et son empreinte environnementale.

Cette analyse a permis de mettre en évidence des performances satisfaisantes en environnement léger, mais aussi des dégradations notables en contexte de forte charge. L'étude environnementale, quant à elle, a révélé plusieurs bonnes pratiques déjà respectées, tout en pointant des vulnérabilités structurelles liées au chargement des ressources, à l'absence de mise en cache, ou encore à l'architecture non adaptée à la montée en charge.

Même en prenant en compte la refonte architecturale du projet et le respect actuel des principes SOLID, une séparation des couches est par défaut de mise et va donc en partie diminuer les performances du site (routing qui mène au controller appelant Service appelant DAO retournant jusqu'au controller qui injecte dans la vue par exemple). Ça n'est malheureusement pas une optimisation que nous pourrons ajuster en ce qui concerne les performances, mais une telle structure nous permettra tout de même d'améliorer les performances plus facilement sur tous les autres aspects du site web.