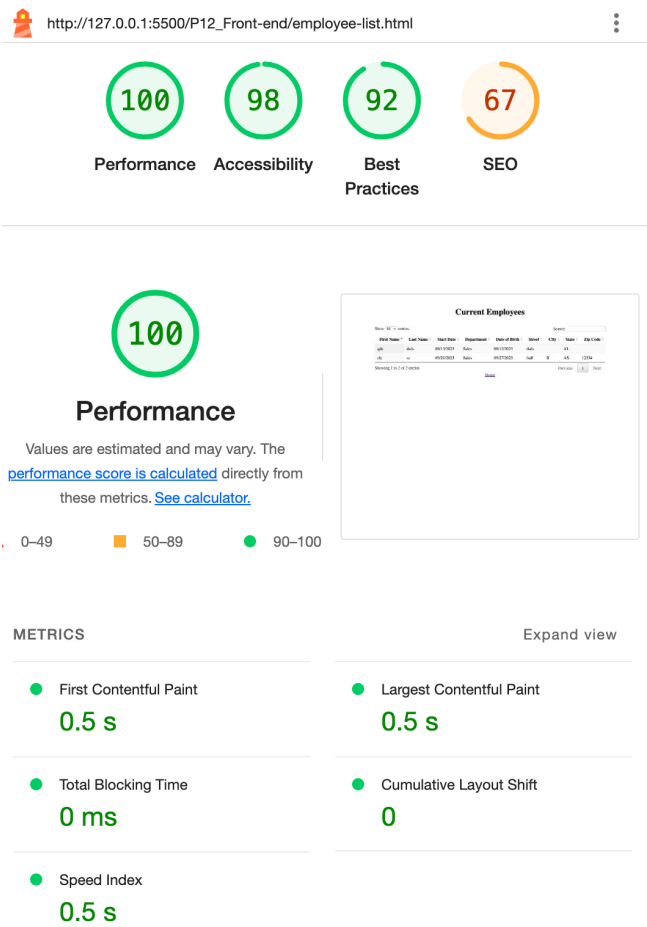


# Rapport sur l'Optimisation des Performances

## Introduction

Le rapport ci-dessous présente les étapes entreprises pour optimiser les performances de l'application React lors de sa migration depuis jQuery. Avant la conversion, l'application a été soumise à un test Lighthouse. Après l'optimisation, un nouveau test Lighthouse a été réalisé, et l'application a obtenu un score de performance de 100 %.

## Jquery App



METRICS

Expand view

● First Contentful Paint

0.5 s

● Total Blocking Time

0 ms

● Speed Index

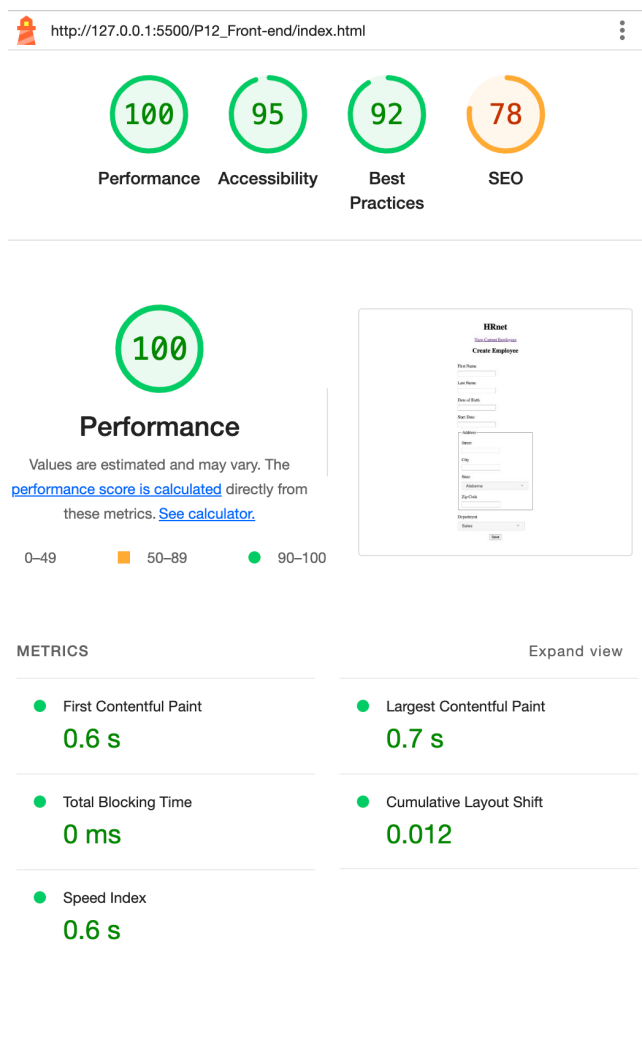
0.5 s

● Largest Contentful Paint

0.5 s

● Cumulative Layout Shift

0



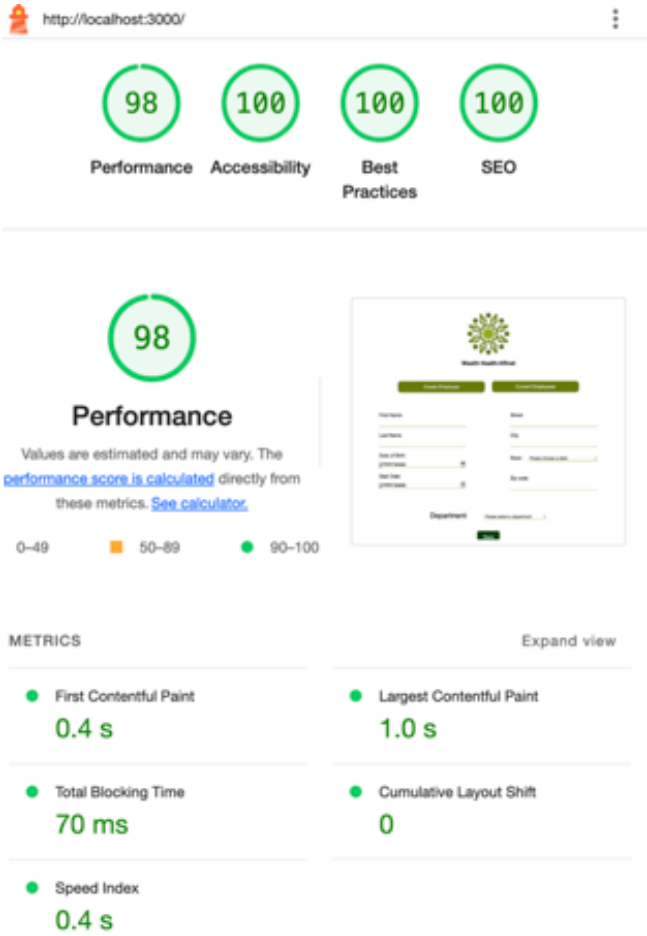
### Remarques sur les resultats:

Certains problèmes ont été identifiés dans l'ancienne application jQuery, notamment

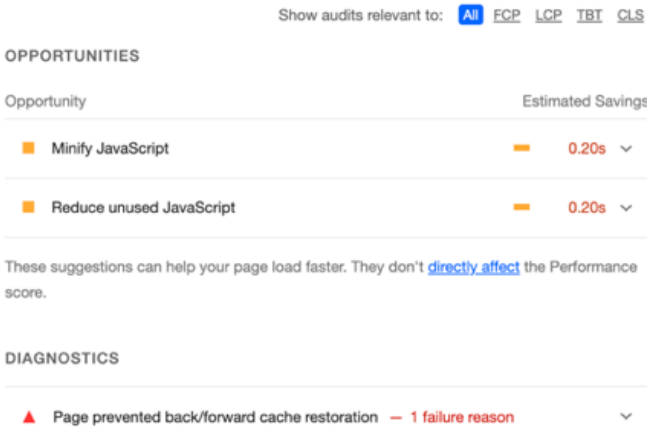
- l'absence de la balise <meta name="viewport">,
- l'absence de l'attribut [lang] dans l'élément <html>,
- l'absence d'une déclaration de doctype dans le document.
- ARIA input fields sont pas des accessible names
- Errors dans les console

# React App

## A- Résultat après conversion et avant optimisation



## B- Amélioration propose par lighthouse pour optimisation



▲	Serve static assets with an efficient cache policy	— 2 resources found	▼
○	Keep request counts low and transfer sizes small	— 5 requests • 612 KiB	▼
○	Largest Contentful Paint element	— 980 ms	▼
○	Avoid long main-thread tasks	— 1 long task found	▼

## Démarches d'Optimisation:

Les étapes suivantes ont été appliquées pour améliorer le score de performance :

- **Code Splitting** : Lors de la génération de la version de production avec Create React App, le processus de build a automatiquement inclus le code splitting pour optimiser la livraison de l'application.
- **Minification du Code** : La minification du code JavaScript a été réalisée pour réduire la taille des fichiers JavaScript, améliorant ainsi les performances globales.
- **Utilisation de memo()** : Le hook React useMemo() a été employé pour mémoriser les résultats de calculs coûteux en performances, évitant ainsi les calculs inutiles lors des rendus de composants.
- **Utilisation de Composants Fonctionnels** : Les composants fonctionnels ont été privilégiés par rapport aux composants de classe pour réduire la charge et améliorer les performances.
- **Analyse et Code-Split des Dépendances** : assurés que les bibliothèques tierces étaient correctement code-split, ne chargeant que les parties nécessaires.
- **Optimisation du Réseau** : Les appels réseau ont été minimisés autant que possible, en utilisant des techniques telles que le prefetching pour charger les ressources à l'avance.
- **Serveur de Fichiers Statiques depuis un CDN** : opté pour la livraison de fichiers statiques depuis un CDN pour une meilleure vitesse de livraison.
- **Optimisation des Web Fonts** : les polices web étaient correctement optimisées, ne chargeant que les variantes nécessaires.
- **Mise en Cache des Données** : Des mécanismes de mise en cache ont été utilisés pour stocker temporairement les données fréquemment utilisées, réduisant ainsi les dépendances aux appels réseau.
- **Élimination du Code Mort** : Tout code inutilisé a été supprimé pour garantir une application légère et efficace.
- **Font-Display: optional** : La propriété CSS font-display: optional; a été utilisée pour optimiser les performances des polices web en permettant au navigateur de décider de leur utilisation immédiate ou de leur chargement en arrière-plan.
- **l'utilisation de \_.debounce()** : techniques de débouclage peut contribuer de manière significative à l'optimisation des performances en réduisant les calculs redondants et en améliorant la réactivité de l'application, en particulier lorsqu'elle est sujette à des interactions fréquentes de l'utilisateur.

## Étape de l'Optimisation sur Accessibilité « score 100% »

- Gestion du contraste des couleurs.
- Test avec des lecteurs d'écran Wave.
- Aria label ajouter

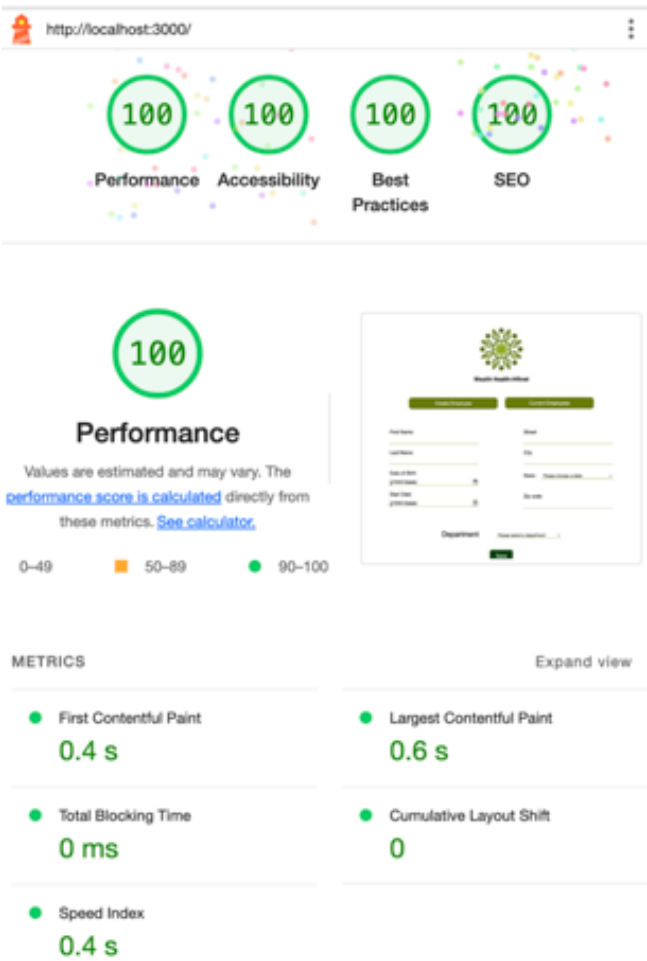
**Étape de l'Optimisation SEO (Optimisation pour les moteurs de recherche) « score 100% »**

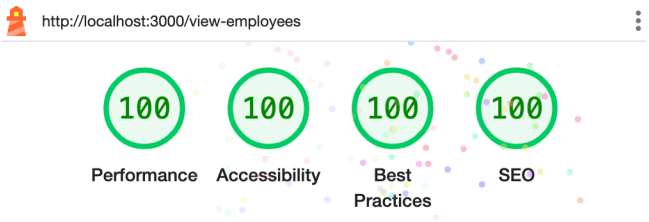
- Recherche de mots-clés.
- Utilisation de balises de titre uniques et des balises de méta-description.
- Optimisation pour les appareils mobiles.

**Étape de l'Optimisation de Best Practice « score 100% »**

- Aucune erreur enregistrée dans la console du navigateur.
- Aucun problème signalé dans le panneau "Issues" des outils de développement de Chrome.
- Utilisation de source maps valides.

**C- Résultat après optimisation**





100

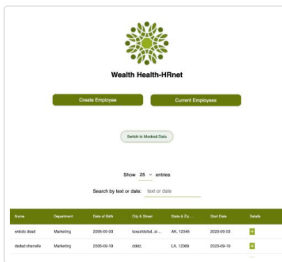
Performance

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)

0–49

50–89

90–100



Page	PageType	Date of Test	File Size	Image Size	Test Date	Score
index.html	Marketing	2020-01-28	400KB	400KB	2020-01-28	100
index.html	Marketing	2020-01-28	400KB	400KB	2020-01-28	100

METRICS

Expand view

● First Contentful Paint

0.4 s

● Total Blocking Time

10 ms

● Speed Index

0.4 s

● Largest Contentful Paint

0.5 s

● Cumulative Layout Shift

0

Comparison

Form Page	FCP	LCP	TBT	CLS	Speed Index
Jquery	0.6	0.7	0	0.012	0.6
React conversion	0.4	1	0	0.7	0.4
React optimisation	0.4	0.6	0	0	0.4

---

## Conclusion

En somme, ces étapes d'optimisation ont permis d'atteindre un score de performance de 100 % pour l'application React, garantissant ainsi une expérience utilisateur fluide et rapide.

