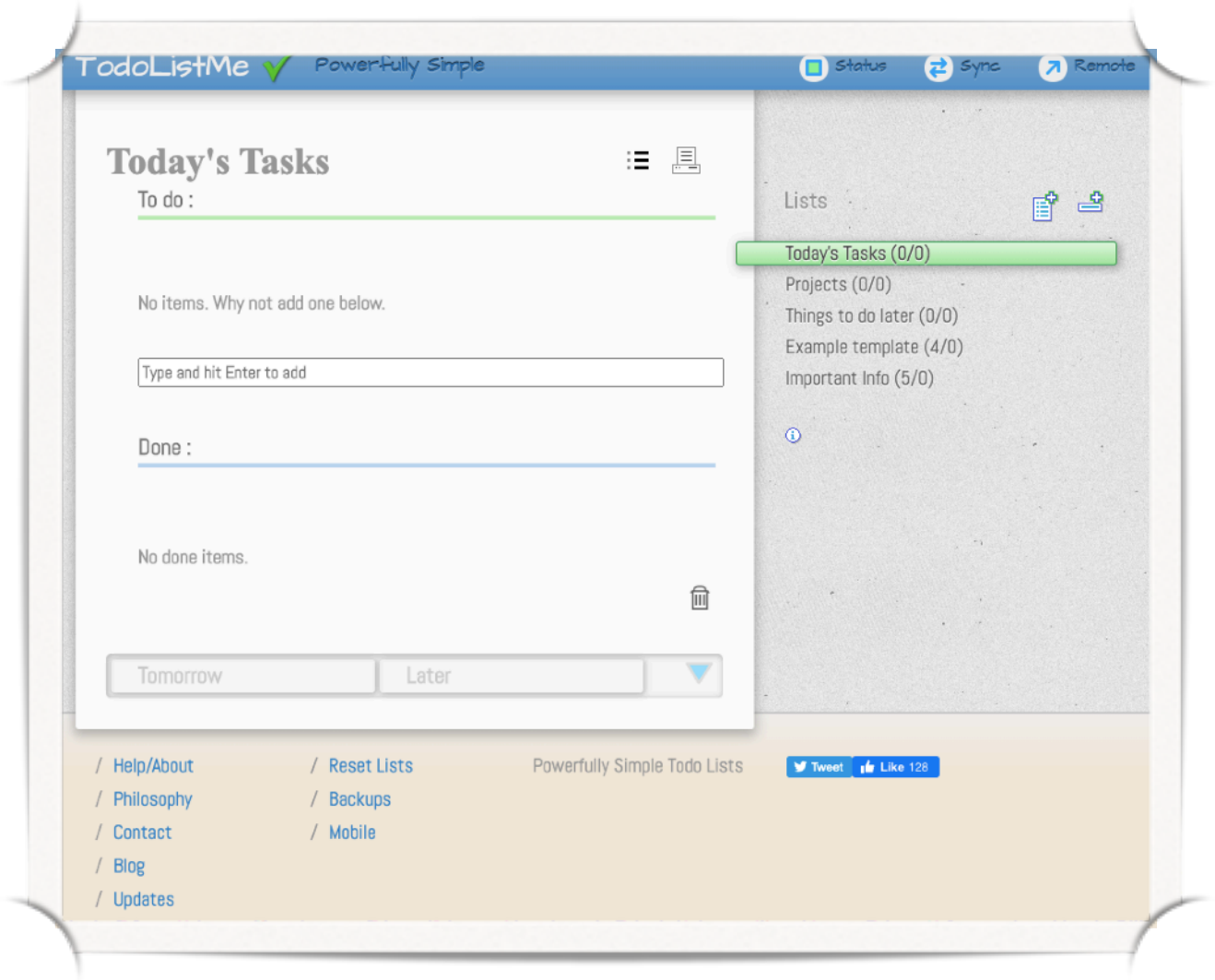
AUDIT
PROJET P8

1. Zoom sur l'application concurrente

L'objectif de cet audit est donc d'avoir une vision d'ensemble des performances de cette application concurrente afin d'optimiser la notre en vue d'un éventuel scaling.

1.1 Présentation.

L'application que nous allons analyser lors de cet audit est TodoListMe. C'est une application qui permet également de lister des tâches. Par contre, elle va plus loin dans les fonctionnalités offertes à l'utilisateur comme par exemple de pouvoir ranger les différentes tâches par catégories.



http://todolistme.net/

1.2 Comparatif entre les deux applications.

	TODOLISTME	TODOS
Rangement par catégorie	✓	✗
Edition de tâche	✓	✓
Programmation de tâche	✓	✗
Publicités	✓	✗
Réseaux sociaux	✓	✗
Stockage	• Local storage • serveur	Local storage

Comme on peut le voir, l'application concurrente possède un nombre de fonctionnalités supérieur à la notre. L'application concurrente à choisit d'implanter de la publicité ainsi que des widget pour les réseaux sociaux. Pour finir, Todolistme propose aux utilisateurs possédant un compte de pouvoir stocker leur liste de tâche sur un serveur. De ce fait, une fois authentifié, l'utilisateur peut retrouver sa liste sur différents appareils.

2. Analyse des performances des deux applications.

Nous allons utiliser l'outil « Network » de la console du navigateur Chrome pour analyser les performances du site concurrent et du notre. Nous allons réaliser 4 tests de performances sur les deux applications, afin d'obtenir des résultats plus affinés. Ces tests ont été réalisés sans l'utilisation du cache.

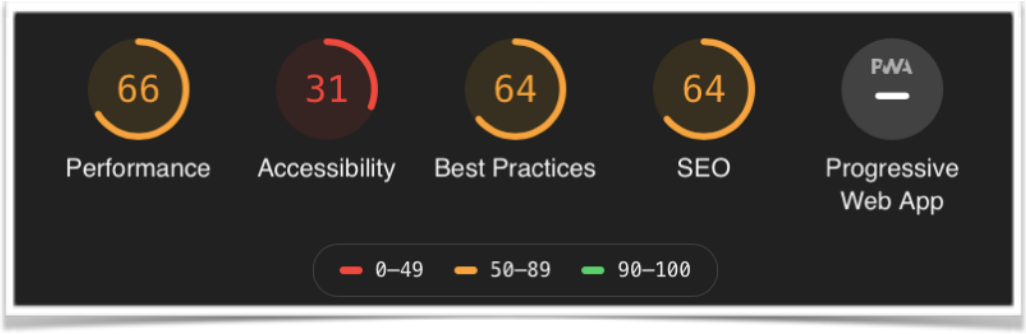
TODOLISTME	1	2	3	4	Moyenne
Requêtes	94	94	94	94	94
Ressources (mb)	3,1	3,1	3,1	3,1	3,1
Téléchargement des ressources (s)	5,97	6,03	6,34	6,07	6,1
Chargement du DOM (s)	1,22	1,43	1,54	1,41	1,4
Chargement de la page (s)	3,47	3,23	3,31	3,42	3,36

TODOS	1	2	3	4	Moyenne
Requêtes	11	11	11	11	11
Ressources (Kb)	27,3	27,3	27,3	27,3	27,3
Téléchargement des ressources (ms)	203	156	176	178	178
Chargement du DOM (ms)	217	188	201	213	205
Chargement de la page (ms)	217	186	199	210	203

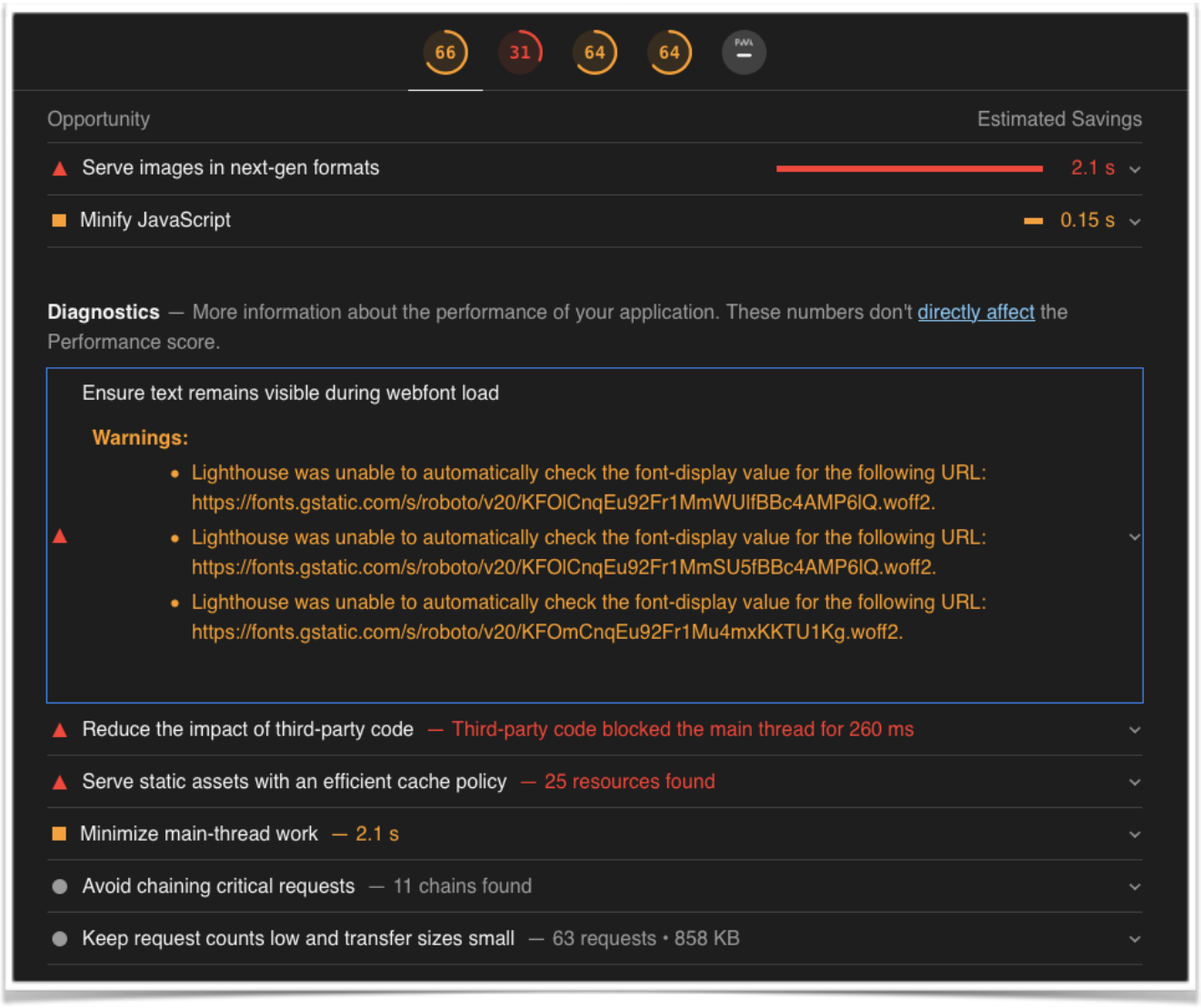
A travers ces deux tableaux, nous constatons que notre application est beaucoup plus rapide. Cela s'explique notamment par un nombre de requêtes moins important que celui de l'application concurrente. Ce résultat est lié au choix de **todolistme** de proposer plus de fonctionnalités. De plus, l'implémentation des réseaux sociaux et le choix d'intégrer de la publicité augmentent considérablement le nombre de requêtes et accentuent le temps de chargement. Le cache permet tout de même de réduire de près d'1/3 le temps de téléchargement des ressources.

Ce qui va être intéressant maintenant c'est de voir ce qui va pouvoir être améliorer. Pour cela, nous allons utiliser l'outil « Lighthouse » de la console du navigateur Chrome.

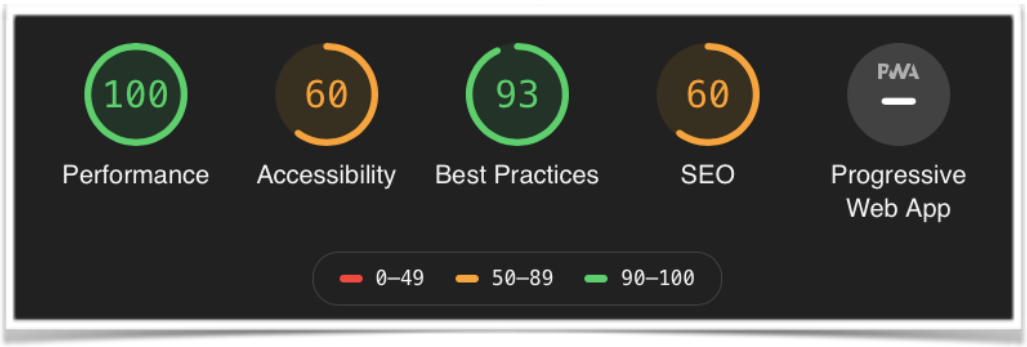
TODOLISTME



Comme on peut le constater ci-dessus, les notes obtenues par todolistme sont plutôt moyennes voir même, mauvaises. Lighthouse nous décrit ci-dessous des pistes d'améliorations en terme de performances.



TODOS



Ci-dessus, les notes obtenues en terme de performance par notre application sont excellentes. Seul bémol au niveau du SEO (référencement) où Lighthouse nous décrit ci-dessous des pistes d'améliorations. Cela n'influe pas directement sur les performances de l'application mais il faudra néanmoins y accorder une importance particulière par la suite.

Conclusions :

Les résultats du site http://todolistme.net/ sont moins performants en raison des critères suivants :

- le site propose plus de fonctionnalités ce qui accentue le temps de chargement,
- Le site n'utilise pas le bon format d'images. Il faut préférer des formats tels que JPG 2000, JPEG XR, plutôt que PNG,
- les fichiers Javascript, CSS et HTML ne sont pas minifiés,
- les fichiers ne sont pas compressés. La compression pourrait permettre d'optimiser le poids de notre code source et contribuer aux bonnes performances de l'application,
- le site à fait le choix d'implémenter des réseaux sociaux (Twitter et Facebook) ainsi que d'intégrer des publicités ce qui augmente le nombre de requêtes et ralentit l'application.

3. Le scaling

Après avoir analyser les performances de cette application concurrente, il va être important d'apporter une attention particulière au points suivants en vue d'un éventuel scaling :

- Minifier nos fichiers HTML, CSS et JS. De nombreux outils permettent d'éliminer les espaces, les sauts de ligne, les commentaires et les délimiteurs de bloc afin d'alléger le fichier au maximum.
- Effectuer une compression de nos fichiers et ainsi optimiser au maximum le poids de notre application. Nous pourrions utiliser un bundler comme webpack par exemple.
- En cas d'ajout d'images, préférer des formats nouvelle génération tels que JPEG 2000 ou JPEG XR.
- Il faut garder en tête que notre application sera plus rapide sans publicités. SI une monétisation de notre application est envisagée à l'avenir, il va falloir trouver un juste milieu afin que la présence de pub n'ait pas d'impact trop négatif sur l'expérience client.
- Réduire les codes tiers afin de limiter les requêtes. On a pu voir que l'implémentation des réseaux sociaux et des publicités pouvait ralentir l'application. On pourrait, par exemple, ne pas les charger dès l'ouverture de la page mais ultérieurement de manière asynchrone ou à la demande du client.
- Et bien sûr, utiliser la mise en cache.