Auteur: Ingrid Anciaux

Audit de performance de notre concurrent todolistme.net

Auteur: Ingrid Anciaux

Sommaire

Auteur: Ingrid Anciaux

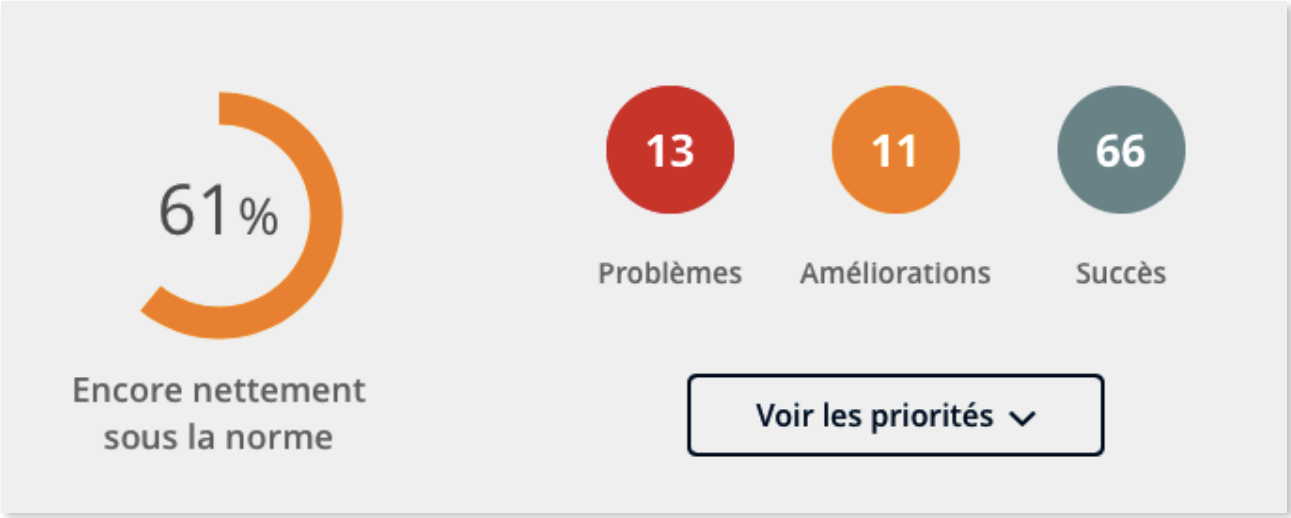
1. Fonctionnement de l'application de notre principal concurrent

Notre concurrent propose une application de todo-list pouvant faire plusieurs choses:
- Créer des listes, des catégories et les réorganiser (Drag & Drop)
- Proposer un système de gestion de calendrier pour fixer des dates d'exécutions
- Possibilité d'éditer, enregistrer et effacer les listes

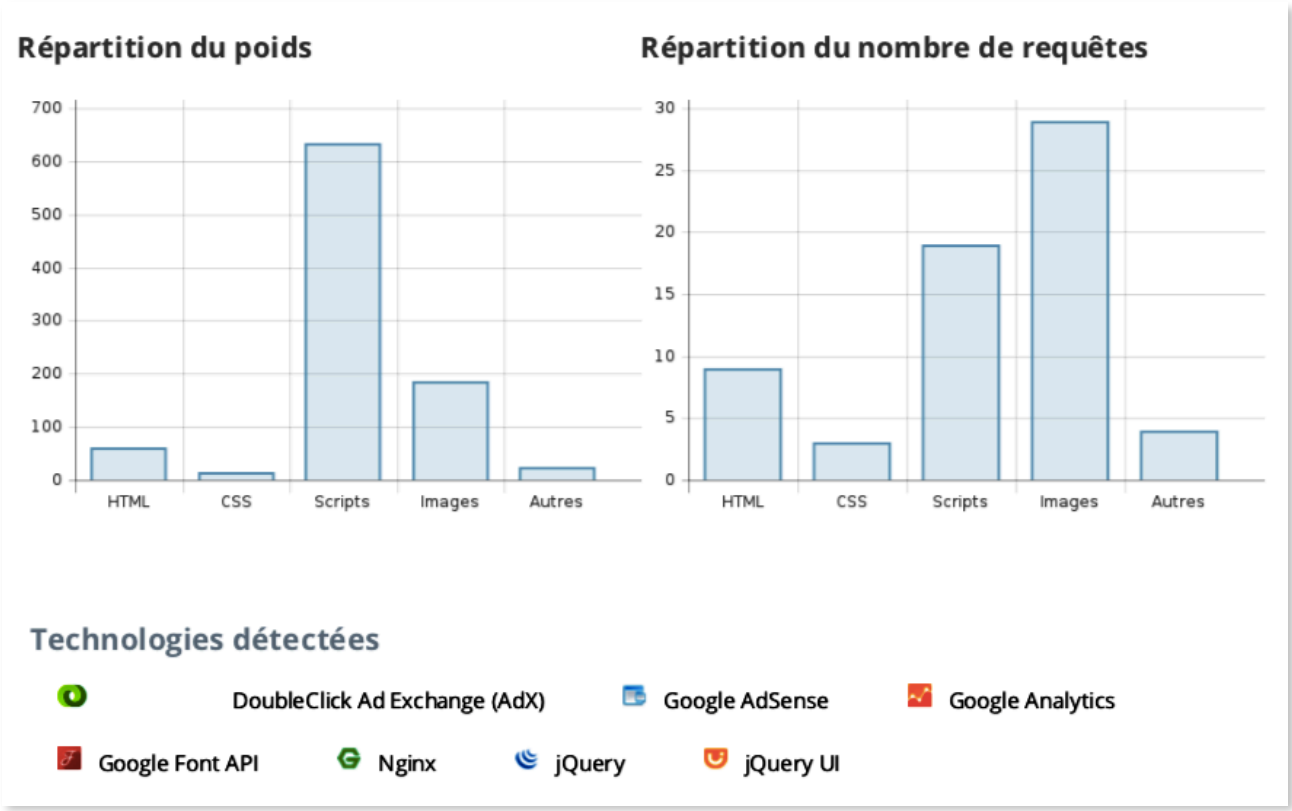
2. Analyse de qualité et performance web

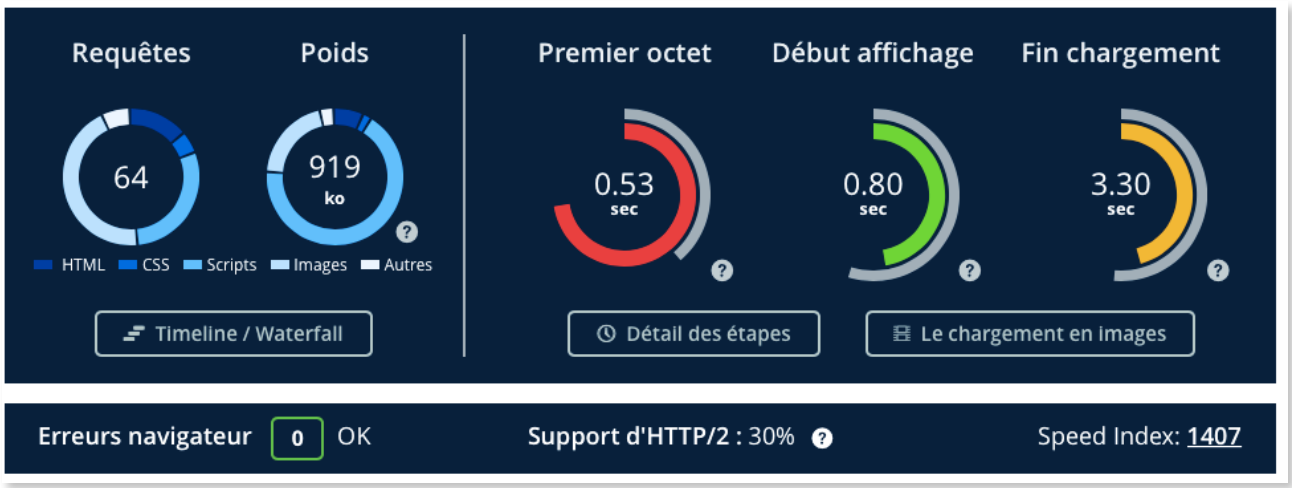
1. En général

Voici un ensemble de tableau bruts que nous allons analyser.

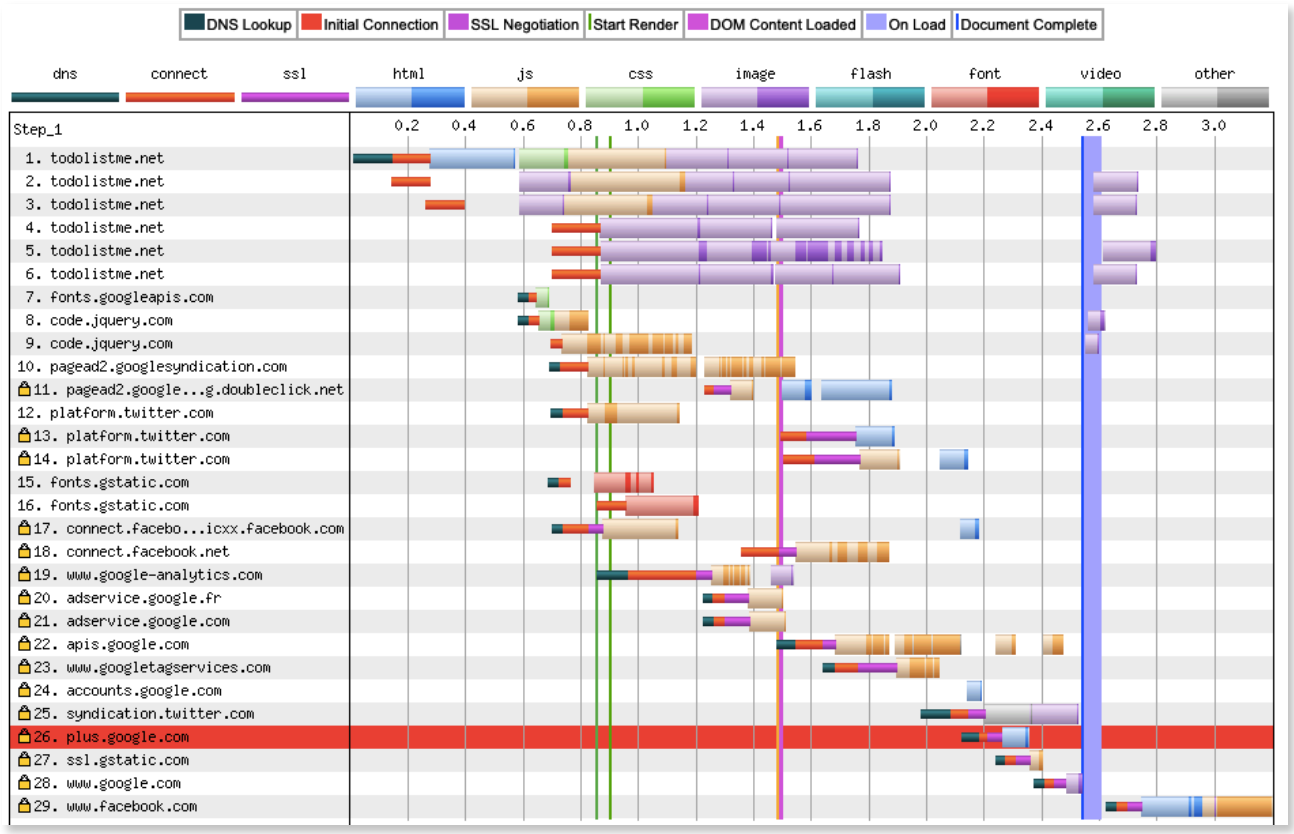


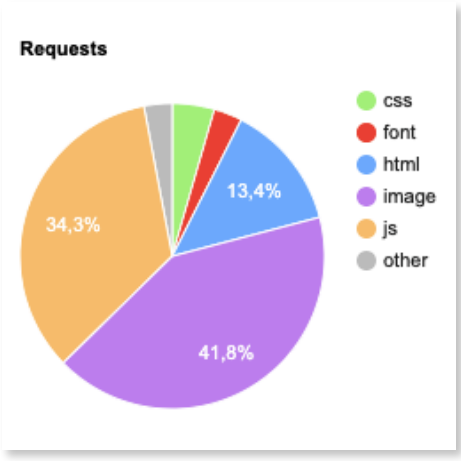
Auteur: Ingrid Anciaux

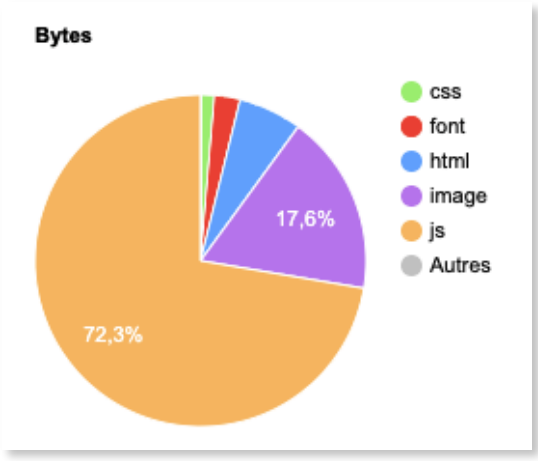




Auteur: Ingrid Anciaux







2. Accessibilité

Les problèmes observés:

- La langue de la page n'est pas spécifiée
- 1 élément vide peut perturber les lecteurs d'écrans

L'optimisation à apporter:

- Expliquer le but de chaque champs de formulaire

Les bonnes pratiques:

- Aucun attribut "src" n'est vide
- Les liens ont des titres cohérents
- Le titre principal est bien le premier titre spécifié
- Tous les labels sont liés à un élément

EN BREF:

La page devrait posséder un attribut `lang` sur le nœud racine `html`: cela permettra aux liseuses d'écran de prendre correctement en charge le site.
Les éléments `<p>`, `<li>`, `<button>`, `<legend>`, `<caption>`, `<figcaption>` and `<quote>` ne doivent pas être vides car s'ils le sont, certains lecteurs d'écran auront des difficultés à interpréter leur présence.
Améliorer l'expérience utilisateur sur votre site en spécifiant une étiquette descriptive pour chaque champ de formulaire serait un plus.

3. Nombre de requêtes

Les problèmes observés:

- 1 ressource est inaccessible

L'optimisation à apporter:

- Economiser 18 requêtes à l'aide de sprites CSS

Les bonnes pratiques:

- Aucunes redirection trop longue n'a été détectée
- Fichiers CSS bien repartis

EN BREF:

Combiner les petites images dans une sprite CSS réduit le nombre de fichiers que le navigateur doit télécharger et accélère donc le chargement de la page de son application.

Chaque requête HTTP représente un coût en terme de performance (roundtrip time, utilisation de bande passante…).

Il est ainsi préférable de faire une requête vers un fichier de 50ko de données plutôt que 10 requêtes vers des fichiers de 5ko.

Les redirections déclenchent des allers-retours évitables sur le réseau, et allongent le temps de chargement de la page.

Aucune ressource n'est pré-chargée sur cette page.
Chaque requête HTTP représente un coût en terme de performance (roundtrip time, utilisation de bande passante...).

4. Qualité

Les problèmes observés:

- Plusieurs éléments utilisent le même identifiant
- Le mot clé !important est utilisé 17 fois

L'optimisation à apporter:

- Séparer les styles du code HTML

Les bonnes pratiques:

- Les images PNG ne sont pas compressées
- Les extensions des ressources sont bonnes
- Toutes les ressources de la page définissent leur type
- Les propriétés CSS ne sont pas dupliquées
- Tous les sélecteurs CSS ne sont pas dupliquées

EN BREF:

Utiliser un même identifiant pour 2 éléments différents peut entraîner des effets de bord, notamment lors de l'exécution de JavaScript ou lors de l'application des règles CSS.

Le mot clé !important s'approche d'un hack permettant d'interdir toute surcharge d'une propriété définie. Si parfois il peut s'avérer utile, cela doit rester exceptionnel !

Des sélecteurs CSS trop complexes nuisent à la lisibilité générale du code et constituent un gain possible en terme de performance.

Dissocier les balises HTML de leur style améliore la maintenabilité du code et favorise sa factorisation.

La compression a un coût, du côté du serveur et du navigateur client. Il ne faut donc l'activer que si elle est efficace.

L'extension d'une ressource permet de repérer facilement le type de contenu qui y est associé. Vous n'avez aucun intérêt à indiquer une extension différente du type réel d'un fichier.

Chaque ressource peut définir le type de son contenu afin de faciliter leur interprétation par les navigateurs web.

Les propriétés dupliquées au sein d'une même règle CSS nuisent à la lisibilité générale du code CSS. C'est également une source d'optimisation, puisque supprimer les éléments dupliqués réduira le poids du fichier.

5. Sécurité

Les problèmes observés:

- Il manque une politique de sécurité sur la provenance des ressources
- Cette page est exposée à des attaques du type "clickjacking"

L'optimisation à apporter:

- Utiliser HTTPS pour collecter des données sensibles
- Bloquer la totalité du contenu lorsqu'une attaque XSS est suspectée
- 3 ressources tierces sont délivrées sans contrôle d'intégrité (SRI)

EN BREF:

La page contient des champs pour récolter des mots de passe ou des informations bancaires, todolistme devrait garantir à l'utilisateur une connexion sécurisée.

Il ne faut pas permettre à des personnes malveillantes d'intégrer les pages de l'application sur leur site.

S'assurer que le navigateur du client fasse son maximum pour prévenir d'une attaque de type XSS.

La page charge des contenus provenant de tiers, il serait donc plus prévoyant de s'assurer de l'intégrité de ces données.

6. Volumes de données

L'optimisation à apporter:

- Réduire certains fichiers JS (minification)
- Alléger la favicon
- Optimiser les images sans perte de qualité

Les bonnes pratiques:

- Les contenus sont bien appelés par des URLs uniques
- La compression des ressources est activée
- Les ressources CSS sont minifiées
- 12 ressources statiques contiennent 288 octets de cookies
- Toutes les ressources ont un poids inférieurs à 1 Mo
- Cette page n'envoie pas trop de cookies
- 25 images utilisent le format PNG de façon adaptée

EN BREF:

En compressant le code JavaScript, on peut libérer de nombreux octets de données et réduire les délais de téléchargement, d'analyse et d'exécution.
En choisissant un format approprié pour les images et en les compressant, on peut libérer de nombreux octets de données.

Une page récupérant une image plus grande que celle réellement affichée charge inutilement un volume de données trop important.

Si un même contenu est référencé plusieurs fois sur la page, il est préférable d'utiliser une seule URL. Cela permet d'éviter de télécharger plusieurs fois ce contenu.

En compressant le code CSS, on peut libérer de nombreux octets de données et réduire les délais de téléchargement et d'analyse.

Les fichiers trop lourds sont à proscrire sur une page web. Il faut s'assurer que tous les éléments du fichier sont nécessaires au chargement de la page.

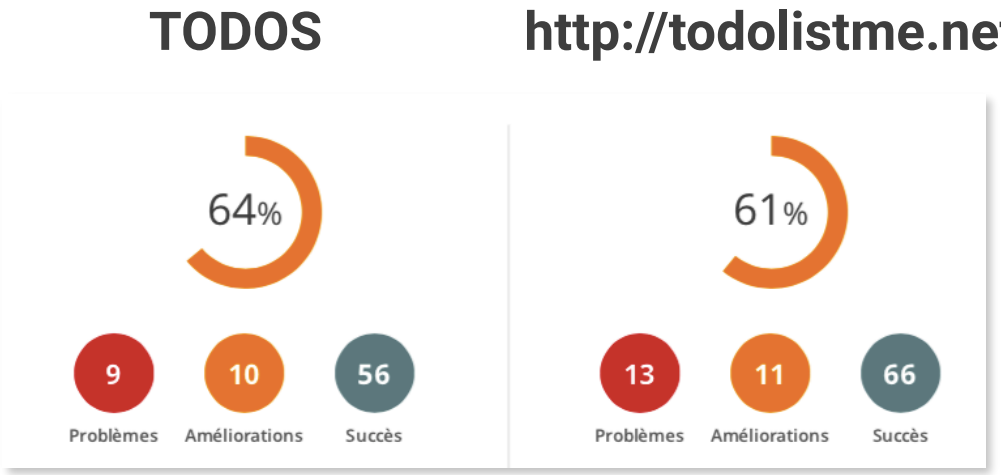
Le contenu d'une redirection ne sera en aucun cas utilisé par les navigateurs web. C'est donc poids inutile qu'il convient d'éliminer.

3. Comparaison

Pour le bien de la comparaison j'ai choisi de mettre le site en ligne sur mon serveur afin d'étudier au mieux sa performance et de proposer la meilleure comparaison possible. Notre application TODOS sera sour URL suivante: ingridanciaux.com/audit

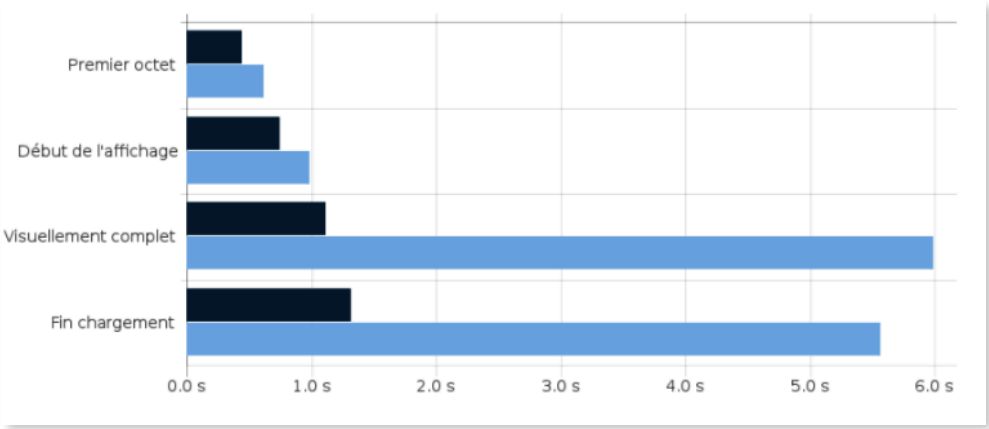
Je vais fournir des graphiques afin de rendre la comparaison simple et plus visuelle. Un audit de performance sera fourni dans la documentation technique afin d'apporter quelques pistes de scaling.

1. Scores de qualité et performance web



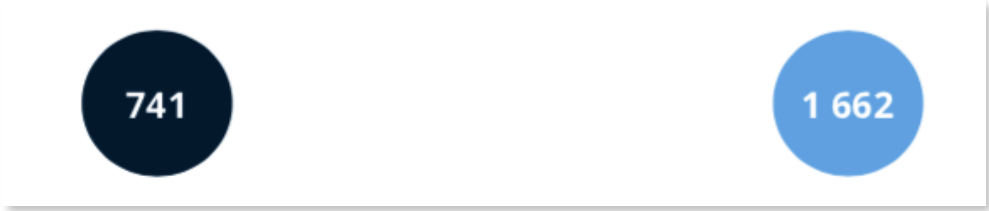
On constate qu'en terme de performance, les 2 applications sont proches. Cela nous amène à penser en premier lieu qu'il y a un effort conséquent à apporter à notre application pour dépasser les 64% de performance et se démarquer de la concurrence pour avoir une application plus performante.

2. Temps de chargement



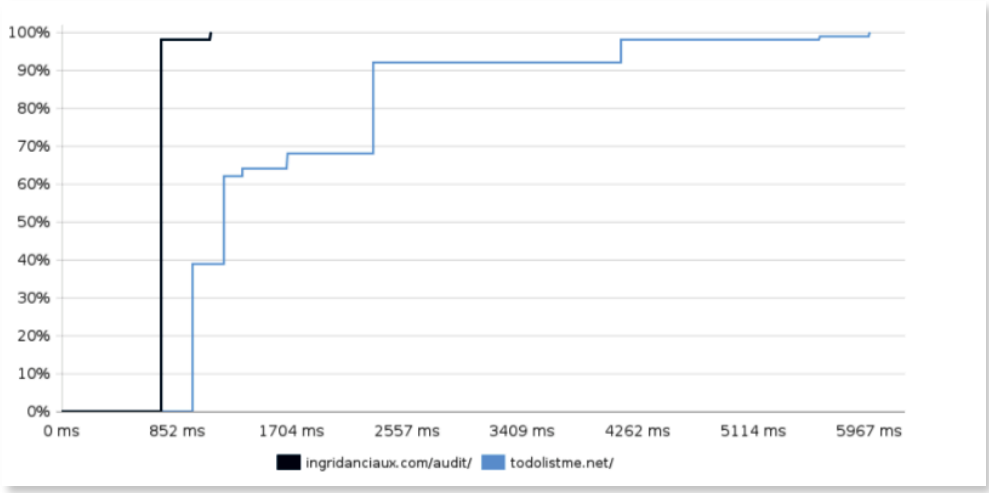
On remarque un gros point fort sur notre application TODOS qui met beaucoup moins de temps se charger. C'est un très bon point.

3. Speed Index



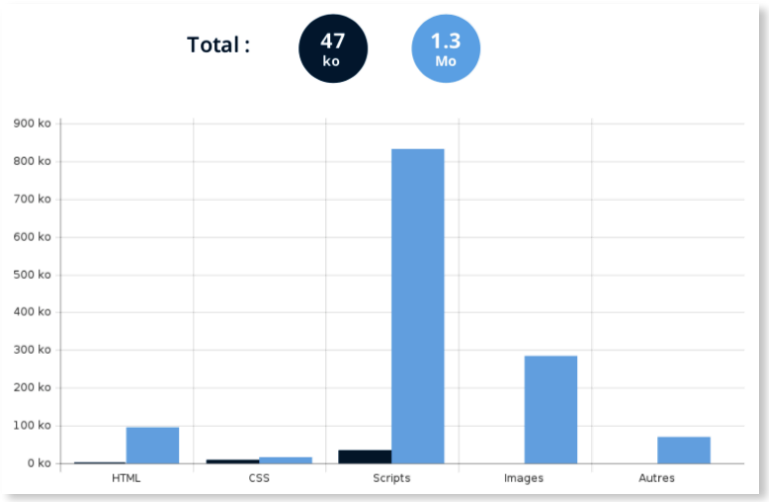
Indice de performance qui retranscrit la vitesse d'affichage de la partie visible sans scroll d'une page web.
Plus l'affichage de cette zone est rapide, plus le speedindex est petit. Recommandation Google : moins de 1000.

4. Progression visuelle



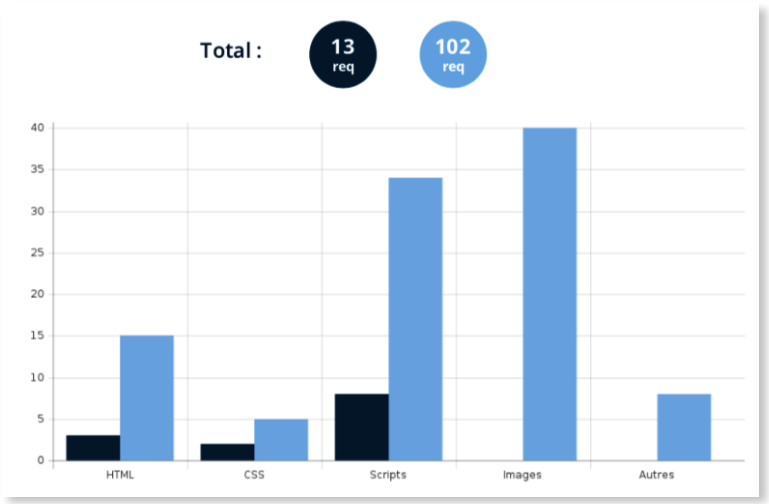
La progression visuelle est très nettement plus rapide sur notre application. Nous atteignons 100% lorsque le site concurrent n'est qu'à 40%

5. Poids par type de ressource



Ce diagramme est très intéressant car on voit nettement une est grande différence globale de poids entre nos 2 applications. Lorsque la nôtre utilise une ressource de 47 Ko, la concurrence en prend 1.3 Mo. Et on voit très clairement qu'ils explosent ce chiffre à cause de leurs fichiers JS. A l'instar, on observe qu'on gagne en ressource du fait que nous n'utilisons pas d'images pour notre application et que notre développement JS en MVC est optimisé.

6. Nombre de requêtes par type de ressources



On reprend ce diagramme pour les requêtes et la différences de requêtes est conséquentes. Nous en faisons seulement 13 contre 102.

4. Conclusion

L'application de notre concurrent todolistme:

- Lente
- Design et user experience a revoir
- Grosses failles observées dans la sécurité
- Trop gros fichier JS

Notre application TODOS:

- Rapide
- Optimiser et performante
- Améliorer l'User experience
- design interessant mais manque de contraste
- une sous catégorie dans les liste pourrait être interessante.