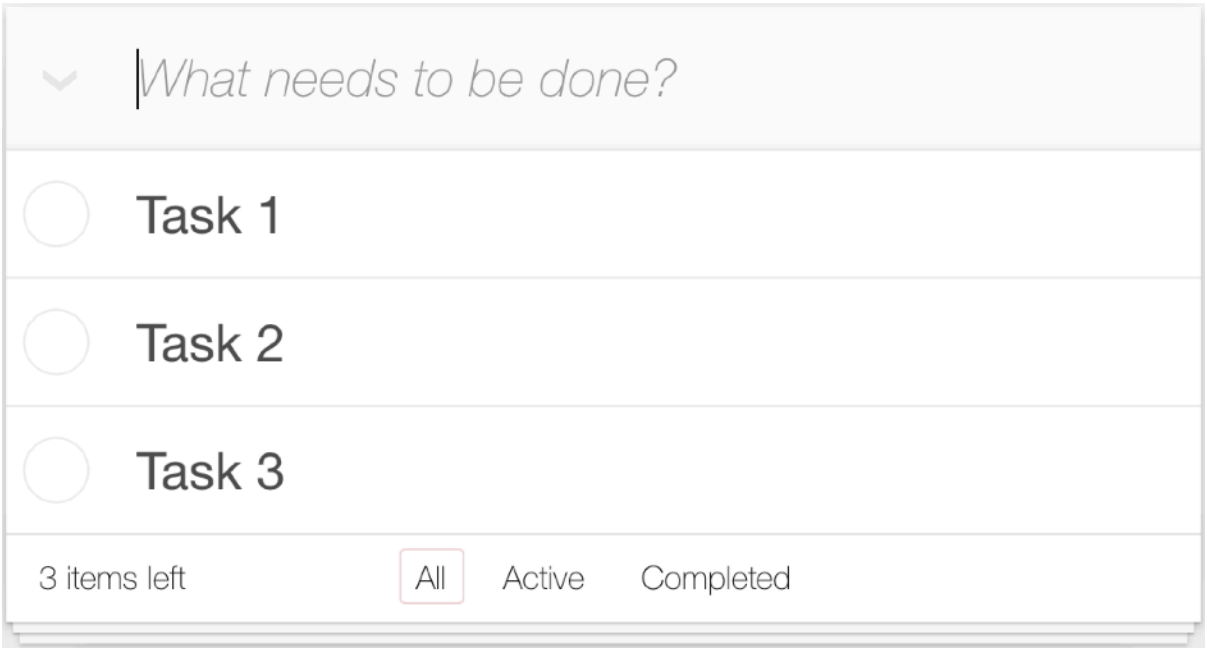
TodoMVC



Audit

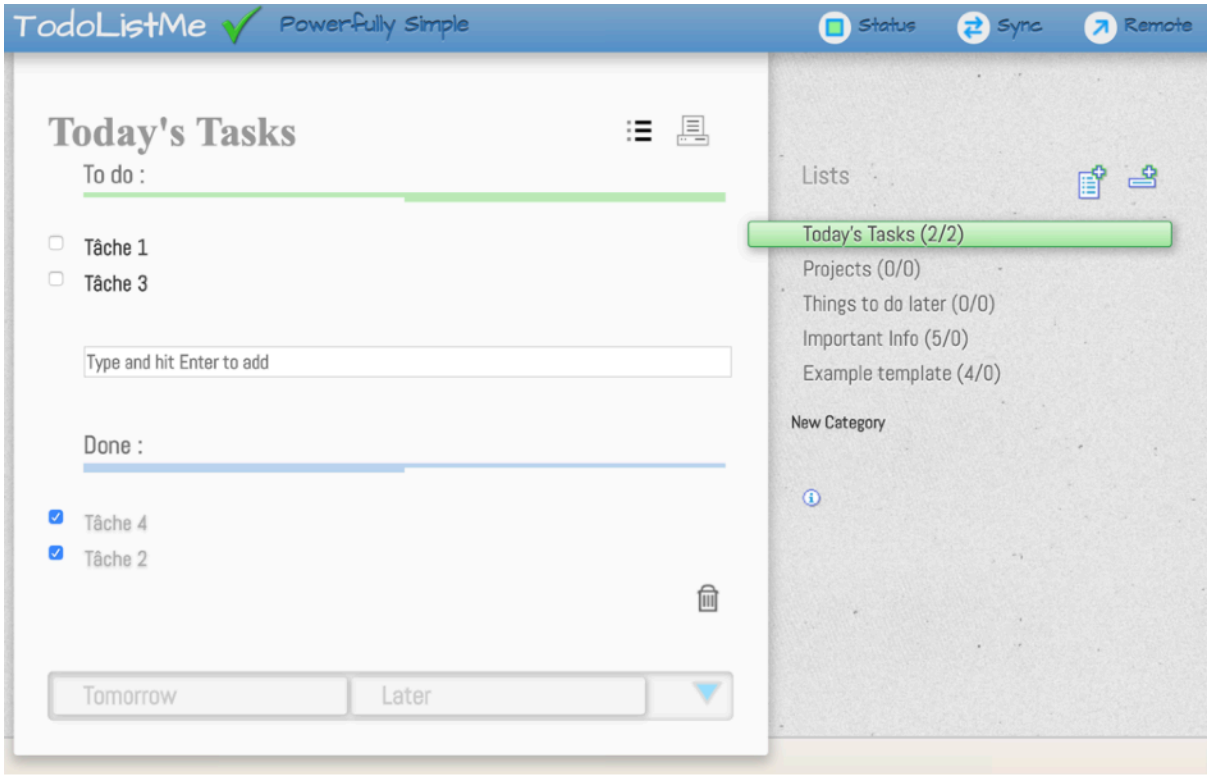
Introduction

Ce document a été crée dans l'optique de dégager des axes d'amélioration pour TodoMVC, une application simple, légère et efficace de création de *to-do list*.
La particularité principale de ce projet est d'être décliné en un grand nombre de *framework* différents, en conservant toujours les mêmes fonctionnalités. Les sources sont librement accessibles, ce qui permet aux développeurs de les aider dans leur choix de framework.
La version étudiée ici n'utilise pas de framework, mais seulement du Javascript dit *vanilla*. Elle utilise le *pattern* **MVC** (Modèle - Vue - Contrôleur).

Cet audit contient le *benchmark* d'une application concurrente, TodoListMe. Les deux applications sont comparées, leur performance est analysée. Sont ensuite présentées les bonnes pratiques que l'on peut tirer de ce benchmark, pour finir sur des pistes d'optimisation d'ordre plus général.

I. Benchmark

L'application concurrente analysée dans ce *benchmark* est **TodoListMe**, dont l'interface est la suivante :



1. Interface utilisateur

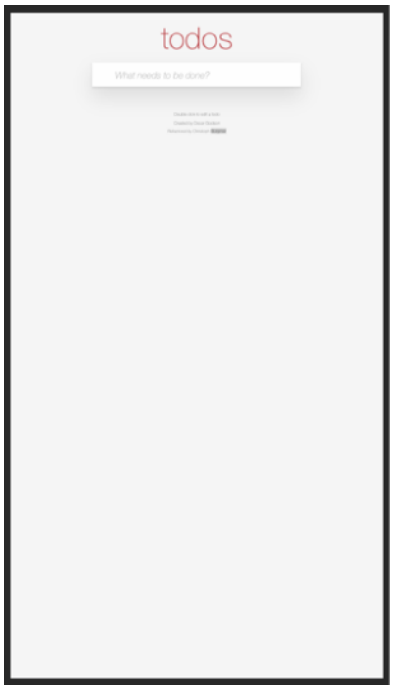
Il s'agit ici du point que l'on remarque immédiatement en comparant les deux applications : **TodoListMe** possède une interface bien plus chargée, moins intuitive aux premiers abords. Cela s'explique par un nombre de fonctionnalités plus grand, mais ces dernières pourraient être mieux agencées et l'interface moins chargée, une application plus fonctionnelle ne devant pas nécessairement empiéter sur l'expérience utilisateur.

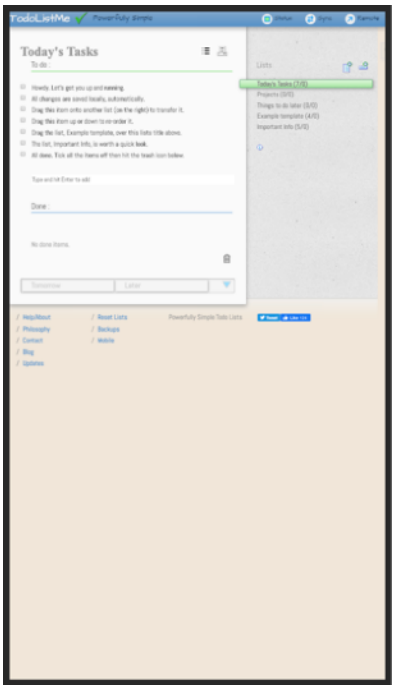
En ce point, TodoMVC est beaucoup plus efficace et va droit au but dans son interface. On comprend en un regard sur l'interface quelles sont les actions à effectuer, la seule explication textuelle étant celle de l'édition de tâche.

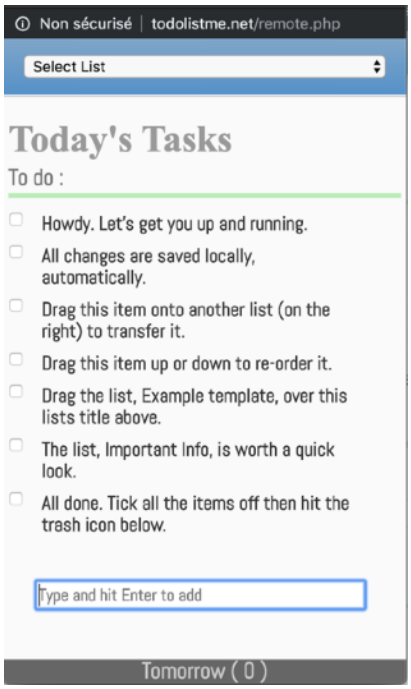
On peut donc dégager le point suivant de cette comparaison : **si l'on enrichit TodoMVC de nouvelles fonctionnalités, il faudra porter beaucoup**

d'attention à l'interface et au design, afin qu'elle garde sa simplicité d'accès et son efficacité.

En terme de *responsive design*, on constate qu'aucune des applications n'est adaptée à un affichage sur smartphone. **TodoListMe** propose néanmoins un affichage « remote » qui ouvre une nouvelle fenêtre avec un affichage adapté aux petits écrans. Si cette solution a le mérite d'exister, elle constitue une mauvaise pratique peu moderne qui est ni optimisée, ni pratique à l'utilisation.
Pour TodoMVC, il faudrait adapter la feuille de style à l'aide de *media queries* pour rendre le site responsive, sans manipulation nécessaire de l'utilisateur.







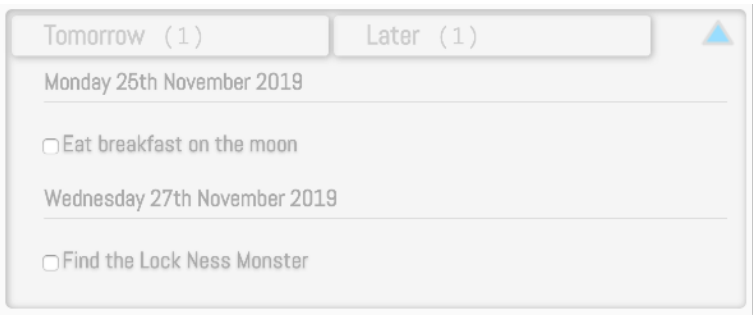
2. Fonctionnalités

Avant de se pencher sur ce qui diffère, on peut tout d'abord poser la base commune aux deux applications : l'ajout de tâches à réaliser, présentées sous forme de liste. Ensuite, ces tâches peuvent être triées, éditées, on peut les cocher pour les déclarer comme « complétées », puis on peut également les supprimer. Les informations sont stockées dans le *local storage* du navigateur.

TodoMeList ajoute d'autres ingrédients à cette recette de base :

- **Tri des tâches :** on peut changer l'ordre des tâches par *drag & drop*, mais aussi par tri alphabétique, aléatoire, ou ne conservant que les trois premières entrées (top 3).

- **Ajout d'une date des tâches :** on peut attribuer des tâches au lendemain, ou à des dates précises). Ces tâches sont ajoutées à une liste classée par date sous la liste principale :



- **Version imprimable :** ouverture d'une nouvelle fenêtre avec une interface adaptée à l'impression.

- **Synchronisation des données :** il est possible de synchroniser ses listes par le biais d'un compte utilisateur. Via ce dernier, l'utilisateur peut envoyer l'état de ses listes sur le serveur, pour les récupérer depuis un autre périphérique. Il a également accès à un historique de ses listes sur les deux dernières semaines.

- **Gestion de listes :** on peut créer de nouvelles listes indépendantes entre elles. Ces listes peuvent également être rangées dans des catégories, également personnalisables.

En parallèle à cela, on constate la présence d'éléments qui ne sont pas liés aux fonctionnalités de l'application :

- **Publicité Google AdSense :** nous pouvons imaginer qu'il s'agit du modèle économique de TodoListMe.
- **Réseaux sociaux :** intégration de partage Twitter d'un lien vers le site, et bouton *like* Facebook de la page de l'application.

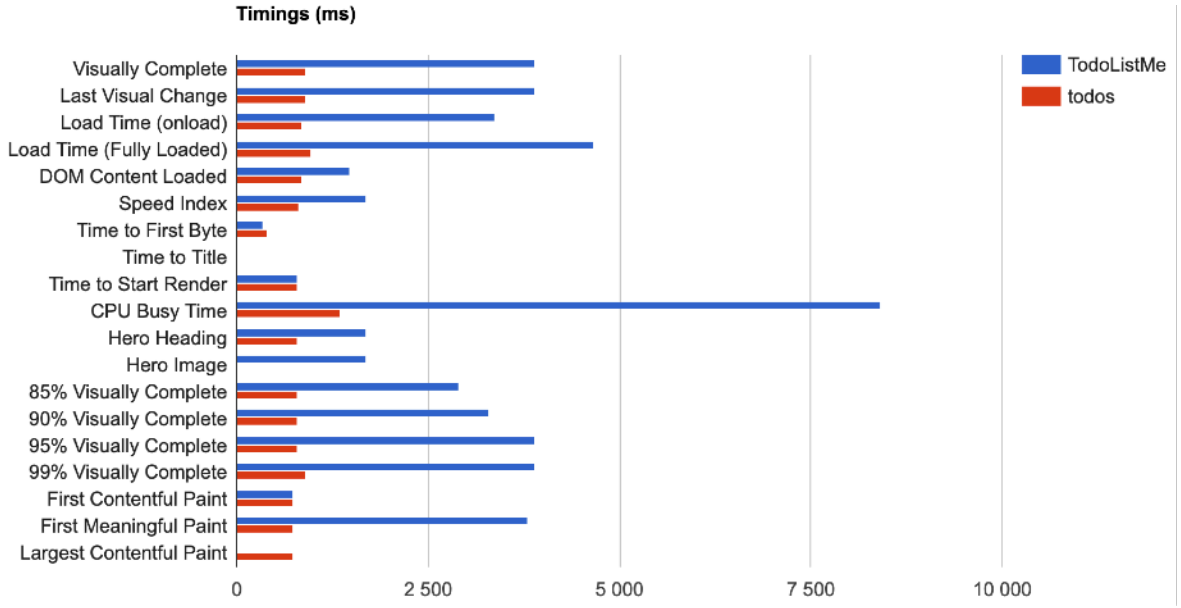
Il faut tout de même nuancer ce nombre de fonctionnalités en plus car les objectifs derrière les deux projets diffèrent légèrement : TodoMVC est une application minimaliste, dont l'esprit est de s'approcher davantage de l'idée d'un post-it collé sur l'écran, simple et immédiat, que d'une application complète de gestion des tâches comme **TodoListMe**. On peut cependant s'inspirer de cette dernière pour en dégager plusieurs pistes d'amélioration pour TodoMVC :

- La possibilité de **drag & drop les tâches** pour les réorganiser permettrait d'ajouter un élément de tri et d'organisation sans pour autant complexifier l'interface.
- La création d'une **feuille de style CSS print** pour permettre une éventuelle impression de la liste, sans alourdir l'application car cela restera transparent pour l'utilisateur.

3. Performance

Toutes les mesures ont été effectuées sur la base d'une connexion d'environ 5 Mbps, ce qui correspond à une connexion ADSL classique.

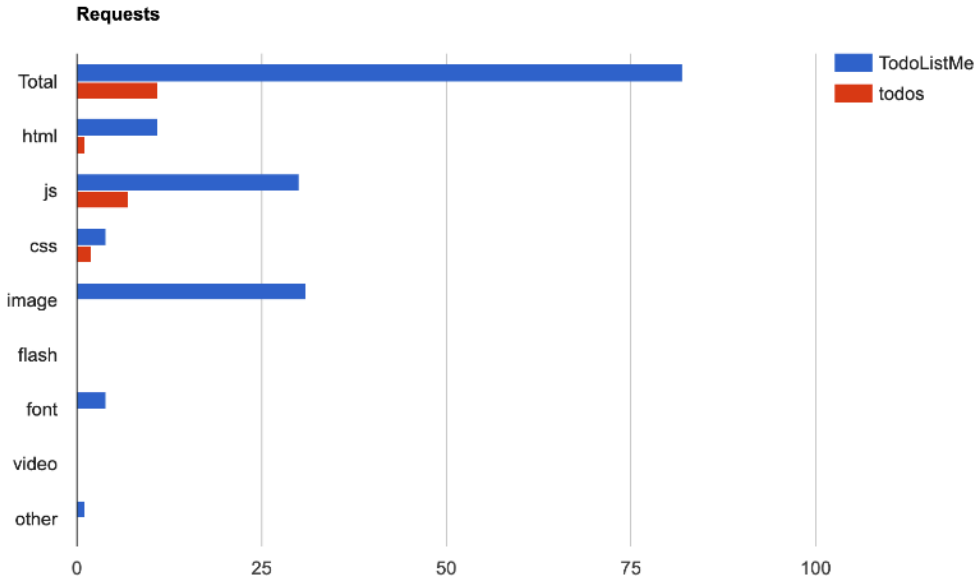
Commençons l'analyse des performances des deux projets par une comparaison des temps de chargement :



Pour un **chargement complet** de l'application, il faut compter environ 4,6 secondes pour **TodoListMe** contre environ 0,97 seconde pour TodoMVC.

Pour être **visuellement complet**, il faut compter environ 3,9 secondes contre 0,9 seconde.

Voyons maintenant le nombre et le type des ressources appelés par les deux applications :



On compte au total **82 requêtes pour TodoListMe**, contre seulement **11 pour** TodoMVC. **TodoListMe** fait appel au total à 30 fichiers javascript, 31 images, 11 fichiers HTML et 4 polices d'écriture.
TodoListMe charge au total environ 1,3 Mo de données, comprenant environ 0,93 Mo de javascript et 0,2 Mo d'images.
De son côté, TodoMVC ne charge que 11 172 octets au total, environ 11 Ko.
Le nombre de requêtes effectuées et le poids des éléments chargés expliquent cette différence de performance entre les deux applications.

Nous pouvons maintenant nous pencher sur le détail des requêtes effectuées par **TodoListMe**.
Les fichiers javascript propres au bon fonctionnement de l'aspect « to-do » de l'application sont les suivants :

jquery-2.2.4.min.js	200	script	(index)	29.6 KB	176 ms
jquery-ui.js	200	script	(index)	122 KB	561 ms
lists.js	200	script	(index)	9.9 KB	316 ms
lib.js	200	script	(index)	1.7 KB	319 ms
javascript_e.js	200	script	(index)	10.1 KB	488 ms

Ensuite, nous constatons que les fichiers les plus longs à charger appartiennent aux catégories suivantes :
- **Les images** : le fichier le plus long à charger de toute l'application (670 ms) est l'image de fond, d'autres images sont également au dessus des 400 ms.
- **jQuery UI** : cette librairie javascript de widgets représente plus de 500 ms de chargement, le fichier de jQueryUI pèse à lui seul 122 Ko.
- Implantation d'une **publicité Google AdSense** et widget **réseaux sociaux Facebook et Twitter.**

Pour plus de détails, voir les tableaux complets des requêtes en annexe.

On peut dégager de ces trois catégories des points de vigilance pour TodoMVC, il s'agit d'éléments qui peuvent beaucoup alourdir une application.

II. Audit

1. Bonnes pratiques

À partir de ce *benchmark*, nous pouvons tout d'abord souligner plusieurs bonnes pratiques à garder en tête pour toute future évolution de TodoMVC :

- **L'intégration d'images** est à limiter, d'autant plus que la fonction du site est avant tout textuelle. En cas d'utilisation d'images, il faut les optimiser au maximum : compression, SVG si possible, *background-repeat* pour le fond, etc.

- **L'intégration de publicité** pose la question de la **monétisation**. De plus en plus d'utilisateurs utilisent des bloqueurs de publicité sur leur navigateur et l'encart pub encombre l'interface du site, en plus d'en alourdir le chargement. D'autres modèles économiques sont envisageables en fonction de l'ambition du projet, comme l'ajout de comptes *premium* avec plus de fonctionnalités. Une inscription (même gratuite) pourrait aussi permettre de récolter des données clients, pour par exemple communiquer avec eux par *newsletters.*

- **L'intégration des réseaux sociaux** est un bon moyen de promouvoir l'application, mais elle doit également se faire de la manière la plus légère et pertinente possible. L'intégration du *widget* complet n'est peut être pas indispensable, un icône avec un lien pouvant suffire à rediriger l'utilisateur vers les réseaux correspondants. On pourrait cependant imaginer un partage de la liste possible, sur Twitter ou Facebook.

- Enfin, **l'utilisation de librairies** comme jQuery/jQueryUI - qui peuvent aujourd'hui être remplacés par du Javascript natif - est à limiter tant que possible, toujours dans l'optique d'alléger l'application.

2. Pistes d'optimisation

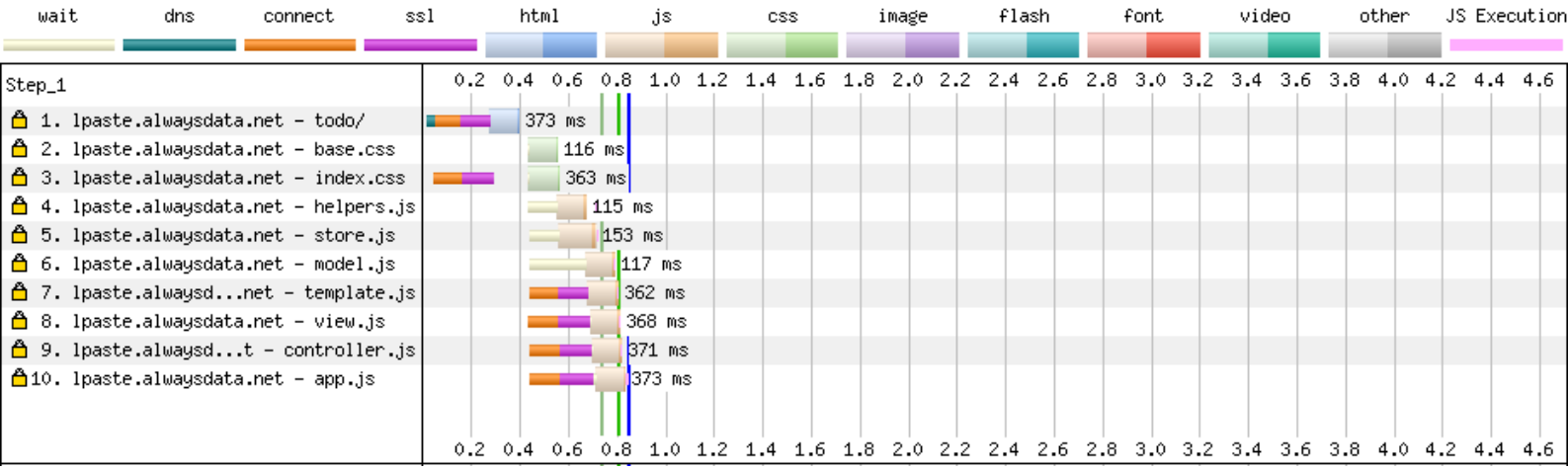
Avec l'aide du site d'analyse Dareboost et des critères « Webperf » de la certification Opquast, voici deux pistes d'optimisation que l'on pourrait appliquer à TodoMVC pour rendre l'application plus qualitative :

- Les fichiers HTML, CSS et Javascript de l'application pourraient être **minifiées** en production, afin de diminuer le volume de données à télécharger.

- Toujours en production, les fichiers Javascript pourraient être regroupés en un minimum de fichiers, afin de créer moins de requêtes quitte à ce que les fichiers soient plus lourds. Un outil comme Webpack pourrait permettre de continuer à travailler en module, tout en permettant la création d'un *build* optimisé pour la production. Un outil comme Babel pourrait également assurer la bonne compatibilité du code sur l'intégralité des navigateurs.

Annexe

1. Tableau des requêtes - TodoMVC



2. Tableau des requêtes - TodoListMe