|  |  |  |
| --- | --- | --- |
| Todolist | Spécification technico-fonctionnelle détaillée | V 1.0 |
| S.L |
| Todolist | 04/05/2019 |



SOMMAIRE

[Guide d’utilisation](#_Guide_d’utilisation)

[Description technique de l’application](#_Description_technique_de)

[Audit de performance](#_Audit_de_performance)

# Guide d’utilisation

## Ajout d’une tâche

L’application todolist permet de créer une liste à puce des tâches que vous avez à faire. En cliquant simplement dans le champ principal, il vous suffira de décrire la tâche et d’appuyer sur la touche « Entrée » et elle s’ajoutera à la liste.

Une fois cela effectué de nouveaux éléments apparaîtront :

* La tâche que vous venez de créer et son statut
* Une puce par tâche pour connaitre leur statut
* Le nombre de tâches restantes
* Le filtre actif parmi les trois disponible (All/Active/Completed)

## Sauvegarde des tâches

L’application sauvegarde les tâches que vous entrez en créant un fichier JSON dans le « local storage » de votre navigateur.

Ce fichier peut être supprimé manuellement par l’utilisateur s’il possède des bases en développement.

## Modification et interaction avec une tâche

En double cliquant sur une tâche vous pourrez modifier son contenu sans avoir à la supprimer et la recréer.

De plus en survolant la tâche vous verrez apparaître une croix qui vous permettra de supprimer cette dernière.

En cliquant sur la puce à gauche de votre tâche vous validerez celle-ci et elle sera donc considérée comme finie (completed) et se verra grisée et barrée et un nouveau bouton apparaîtra « Clear completed ».

## Interaction avec l’application

L’application propose plusieurs interactions avec elle en dehors des tâches.

Premièrement elle dispose de trois filtres cliquables :

* « All » affichera toutes les tâches en cours et finies
* « Active » affichera seulement les tâches en cours
* « Completed » affichera seulement les tâches finies

Si vous avez coché toute vos tâches comme étant finies alors la petite flèche à gauche du champ « What’s to be done » sera highlightée.

Enfin en bas à gauche du cadre de l’application le nombre de tâches restantes sera mis à jour suivant le nombre de tâches en cours.

## Suppression d’une ou plusieurs tâches

Une fois votre tâche effectuée et validée vous pourrez la supprimer en cliquant simplement sur la croix qui apparaît au survol de cette dernière (cf [Modification et interaction d’une tâche](#_Modification_et_interaction)).

Si au moins une tâche possède le statut de « completed » (cf [Modification et interaction d’une tâche](#_Modification_et_interaction)) alors le bouton « Clear completed » apparaitra et sera cliquable. Il vous permet de supprimer d’un coup toutes les tâches que vous avez indiquées comme finies.

Si toutes les tâches ont été supprimées alors l’application reviendra à son format de départ avec le seul champ « What’s to be done » de visible.

# Description technique de l’application

## Présentation du projet

L’application fonctionne sur un motif Modèle-vue-contrôleur ou MVC (Javascript MVC).

C’est un motif d'architecture logicielle destiné aux interfaces graphiques très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

Un modèle (Model) contient les données à afficher.

Une vue (View) contient la présentation de l'interface graphique.

Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur.

## Téléchargement et installation du projet

Vous pouvez télécharger le projet sur ce [dépôt git](https://github.com/SIGIERLuc/todo-list).

À la racine du projet vous trouverez le fichier package.json qui vous renseignera sur les modules dont le projet a besoin pour fonctionner.

Une fois le projet téléchargé et dézippé il faut installer les modules node :

* Rendez-vous à cet endroit dans le projet « Path\todo-list-project\todo-list-app»
* Ouvrez la console et rentrez la commande suivante « npm install »
* Après quelques secondes les modules nécessaires (Path\todo-list-project\todo-list-app\ package.json) seront installés.
* Le projet ainsi que les tests sont maintenant fonctionnels.

N’oubliez pas qu’à la racine un fichier .gitignore est présent et qu’il faudra réinstaller les modules si vous téléchargez sur une autre machine.

## Les différent JS et leurs rôles

Le projet comporte 7 fichiers JS (Path\todo-list-project\todo-list-app\js) indispensables à son fonctionnement (celui présent dans la partie test n’est pas indispensable pour le fonctionnement du projet). Chaque JS à un rôle, le code étant suffisamment commenté, nous n’allons pas décrire ici chaque fonction présente mais faire un résumé de chaque rôle qu’elles remplissent.

### app.js

Va lancer le projet en donnant un nom au fichier JSON qui servira de base de données et initialiser tout le projet.

### store.js

Store va permettre de créer un fichier JSON en local sur le navigateur de l’utilisateur et d’interagir avec ce dernier.

Les fonctions crées ici sont utilisées par model.js afin de créer, mettre à jour, supprimer des todos sur le JSON.

### controller.js

Permet d’interagir avec les actions de l’utilisateur et de modifier les données présentes dans le model et la view.

### model.js

Contient les données et la logique de fonctionnement de l’application. Ce JS va permettre d’interagir principalement avec store.js pour pouvoir ajouter, modifier, supprimer des données.

### view.js

Contient la présentation de l’interface graphique. Elle va donc traduire les fonctions de model.js en éléments du DOM visibles par l’utilisateur.

### helpers.js

Va créer des querySelector et des EventListener afin qu’ils soient réutilisés par view.js pour pouvoir interagir avec le DOM.

### template.js

Crée des templates qui vont être réutilisés dans le code comme l’élément HTML d’une todo ou alors sa construction dans la base de données.

## Les tests

Ici les tests sont réalisés avec le module Jasmine installé un peu plus tôt (cf. [Téléchargement et installation du projet](#_Téléchargement_et_installation) ) et permettent de tester différents cas d’utilisation afin de s’assurer que le code fonctionne correctement.

Les tests en eux même sont écrits en JavaScript dans le fichier ControllerSpec.js et le fichier SpecRunner.html permet de voir si les attentes de chaque test sont justes.

Pour tout ce qui se rapporte au fonctionnement de Jasmine je vous invite à vous documenter sur [la page GitHub de Jasmine](https://jasmine.github.io/index.html)

# Audit de performance

Dans cet audit de performance nous allons étudier notre application, et la comparer à notre concurrent premier. Cela nous permettra de pouvoir améliorer notre site sans reproduire les erreurs commises par ce dernier.

## Les points positif et négatif du concurrent

Le site concurrent est peu sécurisé par l’absence de protocole HTTPS ce qui pose un souci car il propose un moyen d’enregistrement avec adresse Email et mot de passe.

L’absence de balise <noscript> pourrait être un souci si l’utilisateur a désactivé le javascript sur son navigateur ce qui pourrait lui faire quitter le site.

De plus de manière générale le site pourrait être optimisé par le remplacement des images simples par des sprite CSS afin de réduire le poids du site et le nombre de requêtes tout comme l’image monochrome.

Au final l’ajout de publicité augmente le poids total du site ce qui augmente son temps d’affichage.

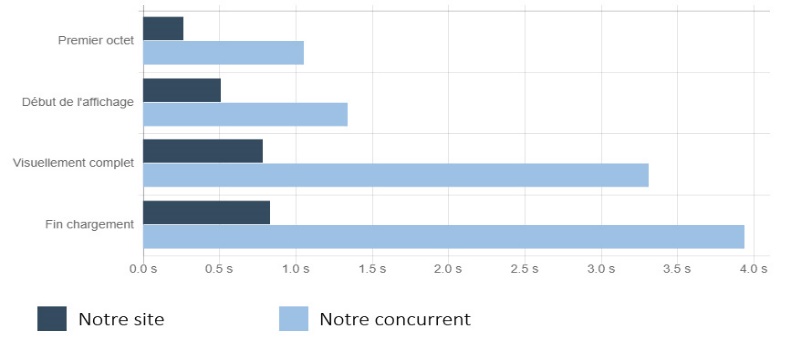
|  |  |
| --- | --- |
| Site : http://www.todolistme.net | |
| Points négatifs | Points positifs |
| Pas de protocole HTTPS | Le nombre de requêtes n’est pas trop élevé |
| Utilisation d’images au lieu de sprite CSS | Les images ne sont pas redimensionnées par le navigateur |
| Deux dépendances critiques |  |
| Pas de balise <noscript> |  |
| Favicon trop lourde |  |
| Une des images monochromes pourrait être remplacée par une class CSS |  |
| L’ajout de publicité |  |

## La comparaison des deux sites dans leur état actuel

Voici un comparatif entre les deux sites :

|  |  |  |
| --- | --- | --- |
| Données | Notre site | Site concurrent |
| Poids | 45Ko | 960Ko |
| Emplacement du serveur | Allemagne | Etats-Unis |
| Vitesse d’accès au serveur | 50ms | 150ms |
| Chargement complet de la page | 0.82s | 3.93s |
| Nombre de requêtes | 13 | 67 (dont 30 images et une dizaine de scripts pour la pub) |

Voici un tableau permettant de voir la progression du chargement dans le temps des deux sites



Avec ces données en tête nous pouvons en déduire certaines choses. Premièrement, bien que notre site soit plus petit et propose moins de fonctionnalités, il est indéniable que le nombre de requêtes ralenti grandement notre concurrent. De plus le choix du serveur impacte grandement le résultat final pour des sites aussi peu lourd.

## Conclusion

Dans l’optique de scaler notre site de todo list sur notre concurrent (<http://www.todolistme.net>) nous devons faire attention à ne pas répéter les erreurs qu’ils ont commises.  
N’oublions pas que dans un souci d’optimisation il serait préférable de crée une base de données externe et de ne plus stocker les todos en « local storage ».

Les points principaux seront de faire attention au nombre de requêtes, à l’utilisation des images et à l’emplacement du serveur. De plus il faut garder à l’esprit de ne pas dépasser les 1Mo pour ce genre de petit site qui doit rester facile et rapide d’accès.