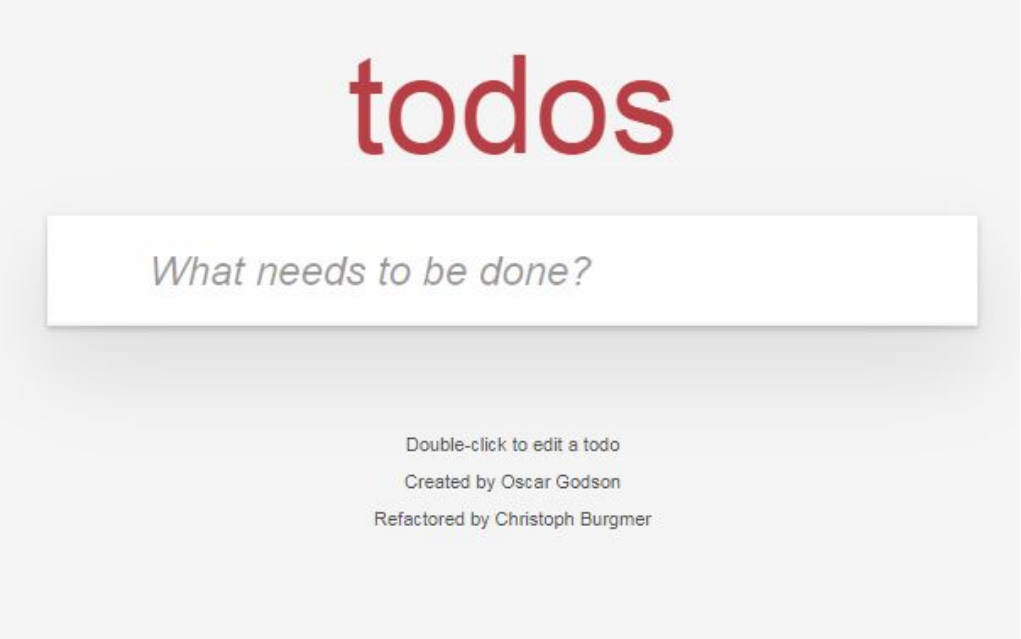
TO-DO LIST

Documentation technique



Version : 1.0

Dernière mise à jour : le 7 mars 2020

Table des matières

1. Introduction

1.1 Généralité

L'application TO-DO List est comme son nom l'indique une liste de tâches.

Notamment utilisée dans le domaine informatique, elle permet de lister des tâches à réaliser ou déjà effectuées, de manière simple et efficace. Ces tâches pouvant être indépendantes ou au contraire être accomplies dans un certain ordre.

Chaque tâche peut à son tour conduire à sa propre liste de sous-tâches, etc.

Dans un travail collaboratif, elles sont associées au collaborateur idoine, et peuvent être divisées ou déléguées.

Cette dernière peut prendre une forme papier (post-it de liste de course) ou numérique (tableur, fichier texte ou logiciel).

1.2 Notre TO-DO List

Dans le cas présent, notre liste de tâche prend une forme numérique au travers d'une application web.

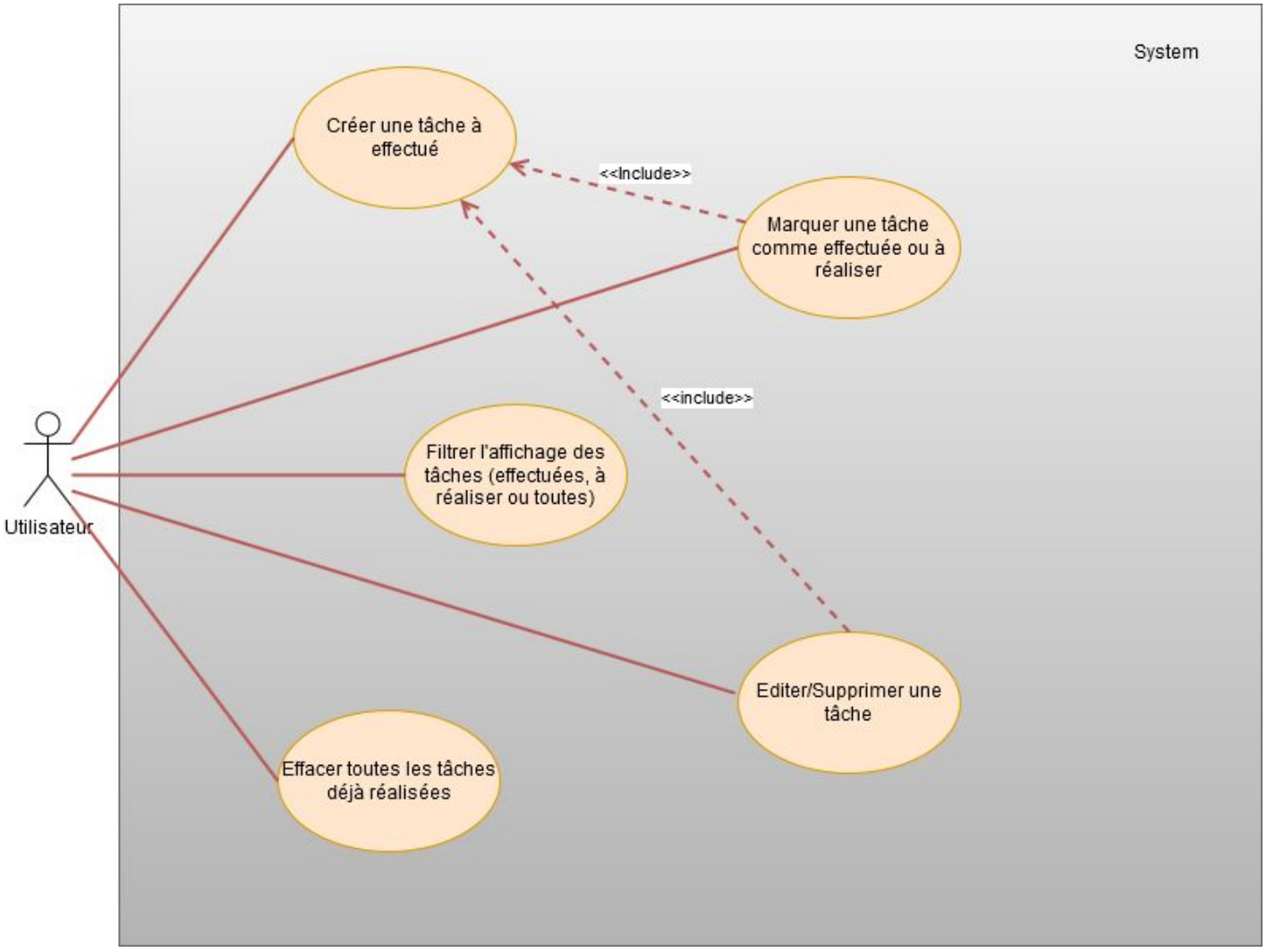
Les listes de tâches étant sauvegardées en local chez l'utilisateur :

- elle ne propose pas d'option collaborative.
- les tâches enregistrés sur un support (smartphone, ordinateur etc) ne sont disponibles que sur ce dernier.

Bien que simple et optimisée, notre API propose l'ensembles fonctionnalités indispensables à une TO-DO List.

Ces dernière sont abordées dans le chapitre suivant.

2. Spécifications fonctionnelles



3. Technologies utilisées

L’application TO-DO List est écrite en Vanilla JS, c’est à dire en pure JavaScript, sans appel à une quelconque bibliothèque. De plus l’API n’utilise aucun framework.

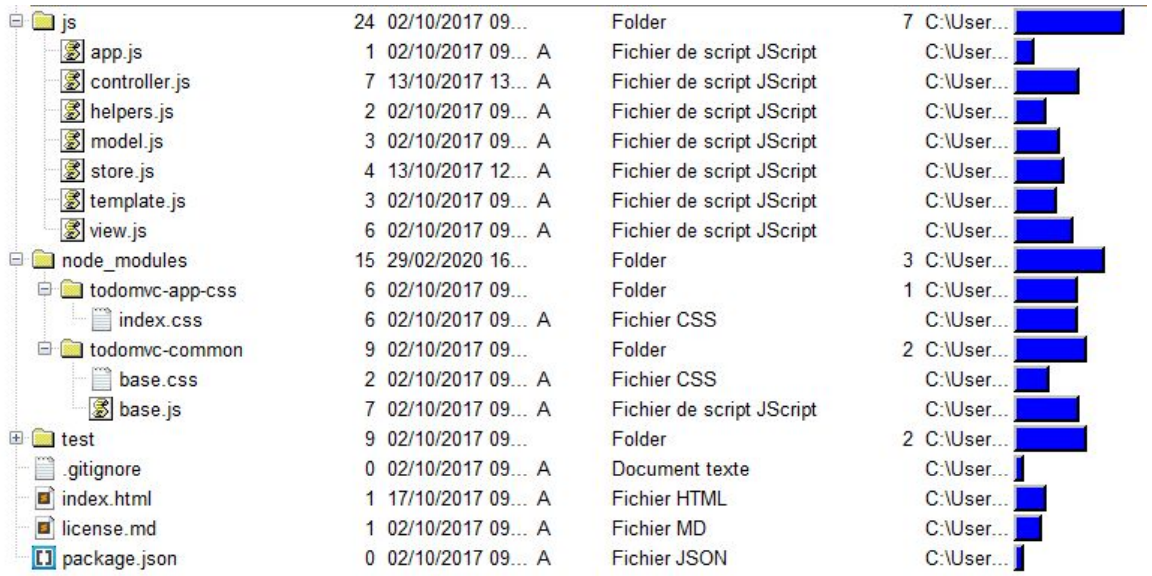
Cette dernière est conforme au motif MVC (Model, View, Controller) ; elle comporte 3 types de modules controller.js, model.js et view.js. Les principes du MVC sont détaillés dans la dernière partie de ce document.

La norme d’écriture utilisée est l'ECMAScript 5.

Cette se veut épurée, simple et rapide, elle ne fait donc appel à aucun services tiers.

Une série complète de tests unitaires et fonctionnels est incluse dans le fichier “ControllerSpec.js”, ce qui permet d’assurer la stabilité de l’application en cas d’ajout futur de fonctionnalités ou de mise à jour du code.

4. Dossiers/Fichiers



app.js	Instantie les classes et appel certaines fonctions au regard d'événements se produisant (ex: chargement du DOM).
controller.js	Aussi appelé code métier : regroupe toute les fonctions répondant aux actions de l'utilisateur.
helpers.js	Contient des fonctions utilitaires.
model.js	Créer une instance d'une tâche et la lies au stockage local.
store.js	Création et gestion de la sauvegarde en local.
template.js	• Prend en charge certains caractères spéciaux et les rends lisibles dans le format HTML. • Gestion des tâches réalisées et du nombre de tâche terminées.
view.js	Gestion de l'affichage du DOM.
Node_module	Contient les fichiers de style css qui sont appelés dans l'index.
Dossier "test"	Définit les tests unitaires et fonctionnels qui seront effectués à l'aide de Jasmine

5. Architecture

L’application utilise un modèle d’architecture MVC (Model View Controller) :

- Contrôleur : Module qui traite les actions de l'utilisateur, modifie les données du modèle et de la vue. Il effectue la liaison entre la vue et modèle.
- Vue: Contient la présentation graphique de l’interface (le DOM) , pouvant capturer les événements, il laisse néanmoins le soin de les gérer au Contrôleur.
- Modèle : Élément qui contient les données ainsi que de la logique en rapport avec les données: validation, lecture et enregistrement.