Application Todo

OC-DAF : Projet 8 par Magalie Huby

Décembre 2020

Présentation de la mission

La mission consiste à poursuivre le développement d'une application 'to-do list '

Les objectifs sont :

- Mettre en Œuvre des tests Unitaires et fonctionnels dans une application web
- Optimiser les performances d'un projet à l'aide des DevsTools
- Reprendre en main un projet Javascript existant

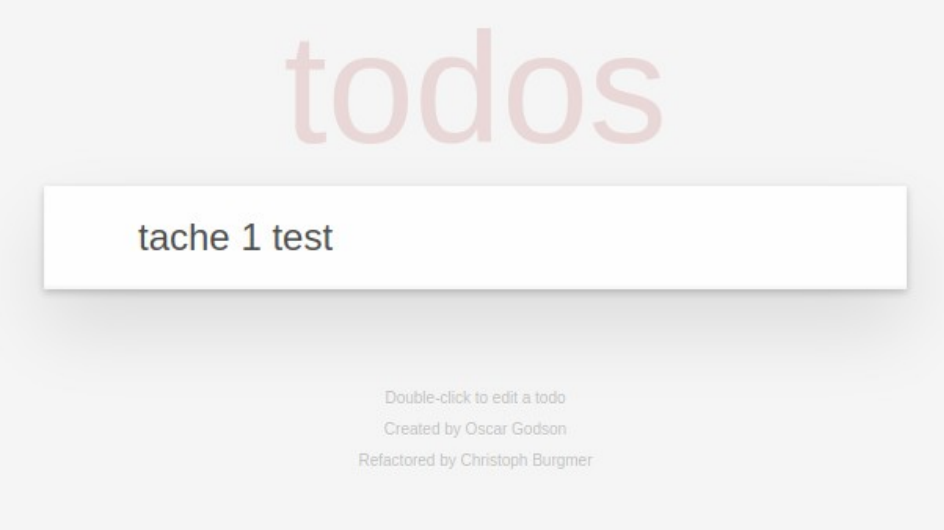
Le projet est décomposé en 4 étapes :

- Étape 1: corriger les bugs
  - Trouver et corriger les deux bugs répertoriés(faute de frappe et conflit éventuel entre deux Id identiques)
  - Optimisation des boucles, affichage d'information dans la console de débogage non nécessaires.
- Étape 2: les tests
  - Ajouter des tests lignes #62, #86, #90, #137, #141, #146, #150, #156, et #196 de ControllerSpec.js .
- Étape 3 : optimiser la performance
  - Audit de performance d'un site concurrent todolistme.net
  - Comparaison avec l'application Todos
  - Axe d'amélioration pour l'application Todos
- Étape 4 : améliorer le projet
  - Documentation utilisateur
  - Documentation Technique
  - Audit

Étape 1 : correction des bugs

Recherche des bugs :

Afin de trouver les bugs donnés dans l'énoncé j'ai testé l'application Todo, et lancé la console de débogage de FireFox.




```
XHR GET http://projet8.initial.hubymagalie.loamok.org/learn.json [HTTP/1.1 404 Not Found 53ms]
GET http://projet8.initial.hubymagalie.loamok.org/favicon.ico [HTTP/1.1 404 Not Found 0ms]
Uncaught TypeError: self.addItem is not a function controller.js:17:9
    Controller  http://projet8.initial.hubymagalie.loamok.org/js/controller.js:17
```

Cela a permis de mettre en évidence, 3 problèmes :

- l'absence du fichier learn.json
- l'abscence de favicon
- un problème concernant la function self.addItem dans le fichier controller.js

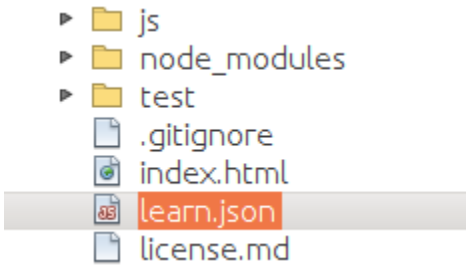
Étape 1 : correction des bugs

→ Absence du fichier learn.json :

On retrouve l'appel du fichier learn.json dans le fichier node_modules/todomvc-common/base.js ligne 248

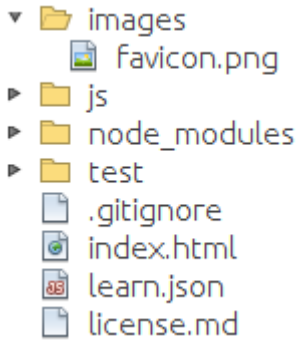
```
                    issueLink.innerHTML = 'This app has ' + count + ' open issues';
                    document.getElementById('issue-count').style.display = 'inline';
                }
            }
        };
        xhr.send();
    }
};

redirect();
getFile('learn.json', Learn);
})();
```

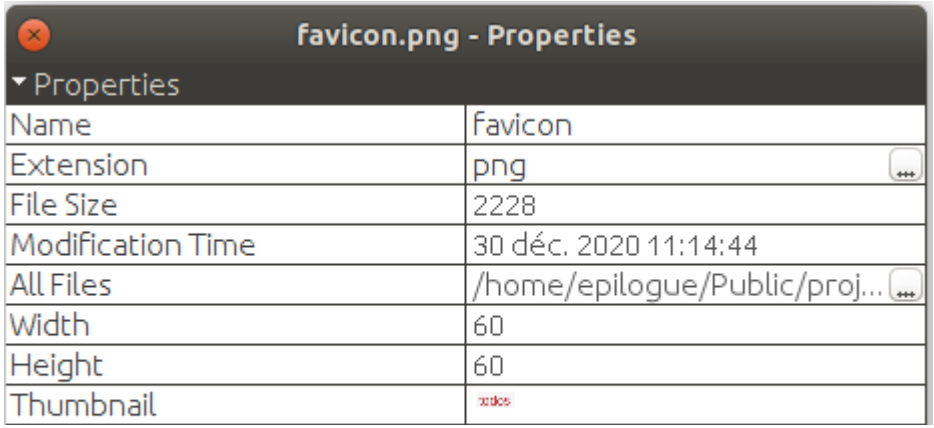
Pour palier à cette absence de fichier, on crée un fichier learn.json vide à la racine du projet



Étape 1 : correction des bugs

→ favicon :
Ajout du dossier images ou sera placé le fichier favicon.png
Fichier qui contient une image réduite de l'application





l'appel se fera dans le fichier index.html au niveau du head

```
<head>
    <meta charset="utf-8">
    <title>To-do list app</title>
    <link rel="stylesheet" href="node_modules/todomvc-common/base.css">
    <link rel="stylesheet" href="node_modules/todomvc-app-css/index.css">
    <!-- HB_19_12_2020 ajout favicon -->
    <link rel="shortcut icon" type="image/png" href="images/favicon.png"/>
</head>
```

Étape 1 : correction des bugs

→ function self.addItem du fichier controller.js :

La fonction est bien appelée dans le fichier controller.js ligne 17

```
        self.view.bind('newTodo', function (title) {
            self.addItem(title);
        });
```

Lorsqu'on la chercher dans le fichier controller.js on la retrouve mais avec une erreur de frappe adddItem au lieu de addItem :

```
/**
 * An event to fire whenever you want to add an item. Simply pass in the event
 * object and it'll handle the DOM insertion and saving of the new item.
 */
// HB_19_12_2020  modification remplacement adddItem par addItem
Controller.prototype.addItem = function (title) {
    var self = this;

    if (title.trim() === '') {
        return;
    }

    self.model.create(title, function () {
        self.view.render('clearNewTodo');
        self._filter(true);
    });
};
```

Étape 1 : correction des bugs

→problème d'unicité d'iD évoqué dans l'énoncé :

La génération de l'Id se fait au niveau du fichier store.js au niveau de la fonction save c'est à dire au moment de l'enregistrement de la tache

```
// Generate an ID
var newId = "";
var charset = "0123456789";

for (var i = 0; i < 6; i++) {
    newId += charset.charAt(Math.floor(Math.random() * charset.length));
}
```

Lors de la génération de l'Id il n'y a pas de vérification par rapport aux Ids déjà existant ce qui peut créer un doublon et donc créé des problèmes lors de l'enregistrement des taches et leurs affichages

Étape 1 : correction des bugs

Solution mis en place :

```
// Generate an ID
var newId = "";
var charset = "0123456789";

// HB_19_12_2020 instanciation de la variable uniqueId à false

var uniqueId = false;
//  HB_19_12_2020 boucle tant que uniqueId est égal à false
while(!uniqueId){
    for (var i = 0; i < 6; i++) {
        newId += charset.charAt(Math.floor(Math.random() * charset.length));
    }
// HB_19_12_2020 condition de sortie du while
    uniqueId = true;

// HB_19_12_2020 boucle pour comparer l'Id généré aux Id déjà présents
    for (var i = 0; i < todos.length; i++) {
// HB_19_12_2020 si l'Id existe déjà  alors on relance la génération d'un Id
        if (todos[i].id === newId) {
// HB_19_12_2020  si newId identique à Id existant on repasse uniqueId à false pour refaire un tour de while
            uniqueId = false;
        }
    }
}
```

Étape 1 : correction des bugs

Optimisation des boucles et affichage d'information dans la console de débogage non nécessaires :

→ optimisation boucle :

Dans le fichier store.js au niveau de la fonction remove ligne 122

```
Store.prototype.remove = function (id, callback) {
    var data = JSON.parse(localStorage[this._dbName]);
    var todos = data.todos;
    var todoId;

    for (var i = 0; i < todos.length; i++) {
        if (todos[i].id == id) {
            todoId = todos[i].id;
        }
    }

    for (var i = 0; i < todos.length; i++) {
        if (todos[i].id == todoId) {
            todos.splice(i, 1);
        }
    }

    localStorage[this._dbName] = JSON.stringify(data);
    callback.call(this, todos);
};
```

La boucle for est dupliquée et pourrait être ramenée à une seule boucle avec les deux conditions l'une après l'autre

Étape 1 : correction des bugs

Ce qui nous donne :

```
Store.prototype.remove = function (id, callback) {
    var data = JSON.parse(localStorage[this._dbName]);
    var todos = data.todos;
    var todoId;
    // HB_19_12_2020 suppression d'une des boucles for  qui était identique à la première
    for (var i = 0; i < todos.length; i++) {
        if (todos[i].id == id) {
            todoId = todos[i].id;
        }
        if (todos[i].id == todoId) {
            todos.splice(i, 1);
        }
    }

    localStorage[this._dbName] = JSON.stringify(data);
    callback.call(this, todos);
};
```

→ affichage d'information dans la console de débogage non nécessaire :

Dans le fichier controller.js au niveau de la fonction removeItem ligne 165

```
    items.forEach(function(item) {
        if (item.id === id) {
            console.log("Element with ID: " + id + " has been removed.");
        }
    });
```

Cette partie de code a été mis en commentaire

Étape 1 : correction des bugs

Ce qui nous donne :

```
/** HB 19_12_2020  console.log qui traine
*       items.forEach(function(item) {
*           if (item.id === id) {
*               console.log("Element with ID: " + id + " has been removed.");
*           }
*       });
*/
```

Boucle while dans le fichier controller.js ligne 120

```
Controller.prototype.editItemSave = function (id, title) {
    var self = this;
            /*HB 19_12_2020 remplacement des deux  whiles  par une regex*/
            title = title.replace(/^\s*/, '').replace(/\s*$/, '');
//      while (title[0] === " ") {
//          title = title.slice(1);
//      }

//      while (title[title.length-1] === " ") {
//          title = title.slice(0, -1);
//      }
```

Étape 1 : correction des bugs

Autres Modifications :

→ Dans le fichier index.html

```
<section class="main">
    <input class="toggle-all"  type="checkbox">
    <label for="toggle-all" >Mark all as complete</label>
    <ul class="todo-list"></ul>
</section>
```

Le label for = '' toggle-all" n'a pas de référence possible à l'input. Ajout de id ="toggle-all' à l'input.

```
<section class="main">
    <input class="toggle-all"  type="checkbox" id="toggle-all">
    <label for="toggle-all" >Mark all as complete</label>
    <ul class="todo-list"></ul>
</section>
```

Étape 1 : correction des bugs

Modification du fichier index.css :

pour des raisons de visibilité, accessibilité mis en place de contraste plus élevé.

ligne 6

ajout background pour le contraste

ligne 7

ajout color:#000000; pour la lisibilité

ligne 30

commentée remplacée par la ligne 31

ligne 59

commentée remplacée par ligne 60

ligne 66

commentée remplacée par ligne 67

ligne 73

commentée remplacée par la ligne 74

ligne 145

commentée remplacée par la ligne 146

ligne 151

commentée remplacée par la ligne 152

ligne 224

commentée

ligne 230

commentée remplacée par ligne 231

ligne 247

commentée

ligne 251

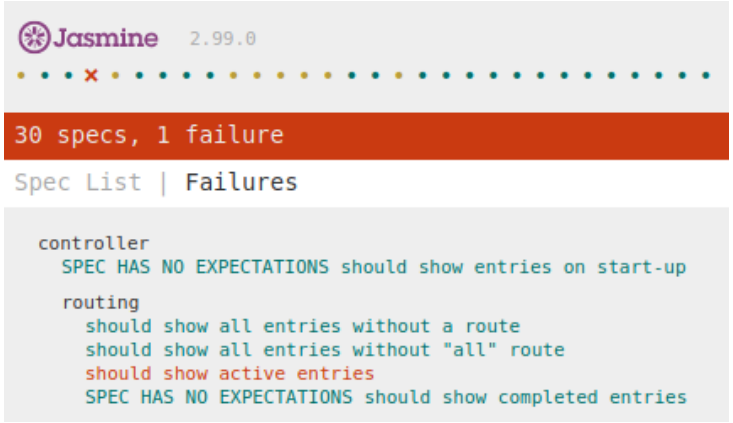
commentée remplacée par ligne 252

Étape 2 : les tests

Écrire les tests :

Utilisation du principe du TDD on écrit d'abord des test qui échoue pour ensuite écrire le test qui réussi .
Exemple1 test "should show active entries" ligne du fichier controllerSpec.js :
fait référence à la méthode showActive ligne 49 du fichier controller .js
Étape 1 le test qui échoue :

```
it('should show active entries', function () {
    // TODO: write test
                /* HB_26_12_2020 doit afficher toutes les taches actives donc qui ne sont pas completed */
                /* 1 on écrit un test qui échoue  donc avec le paramètre completed à true
                 * puisque les entrées avec la route active on le paramètre completed à false */
                /*instanciation de la variable todo */
                var todo = {completed:true};
                setUpModel([todo]);

                subject.setView('#/active');
                /* passage du paramètre attendue pour que le test fonctionne  completed:false */
                expect(view.render).toHaveBeenCalledWith('showEntries', [{completed:false}]);

});
```



Étape 2 : les tests

Étape 2 : le test qui réussi

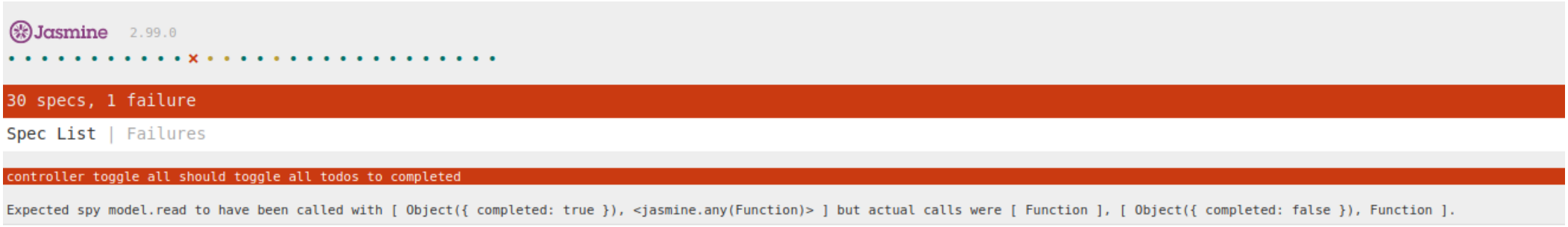
```
it('should show active entries', function () {
    // TODO: write test
                /* HB_26_12_2020 doit afficher toutes les taches actives donc qui ne sont pas completed */
                /* 1 on écrit un test qui échoue  donc avec completed à true
                 * puisque sa route sera /completed  et que l'on cherche toutes les entrées avec la route active*/
                /*2 après vérification que le test a bien échoué
                 * on peut instancier la variable todo  avec la bonne valeur pour le paramètre completed*/
                /*instanciation de la variable todo */
                var todo = {completed:false};
                setUpModel([todo]);

                subject.setView('#/active');
                expect(view.render).toHaveBeenCalledWith('showEntries', [{completed:false}]);

});
```

```
30 specs, 0 failures

controller
  SPEC HAS NO EXPECTATIONS should show entries on start-up

  routing
    should show all entries without a route
    should show all entries without "all" route
    should show active entries
    SPEC HAS NO EXPECTATIONS should show completed entries

  should show the content block when todos exists
  should hide the content block when no todos exists
  should check the toggle all button, if all todos are completed
```

Étape 2 : les tests

Exemple2 test "should toggle all todos to completed " :
fait référence à la méthode toggleAll ligne 219 du fichier controller .js

```
it('should toggle all todos to completed', function () {
    // TODO: write test
                /* HB_26_12_2020 doit mettre toutes les taches a completed
                 * créer deux deux taches donc paramètre completed à false
                 * lancer le trigger pour l'événement toggleAll  pour modifier le paramètre completed à true pour les 2 taches
                 * vérifier que le model lise bien les 2 taches en completed false en phase initial
                 * vérifier que le model  met bien à jour les deux taches avec  le paramètre completed à true
                 * test qui échoue vérification du model avec completed false
                 * */
                /*initialisation des deux taches*/
                var todos = [{id:123456,title:'tache1',completed:false},
                    {id:234561,title:'tache2',completed:false}];
                setUpModel(todos);
                subject.setView('');
                view.trigger('toggleAll',{completed:true}
                );
                /* vérification que le model lit bien les taches avec completed à false
                 * test qui échoue avec completed à true
                 */
                expect(model.read).toHaveBeenCalledWith({completed: true}, jasmine.any(Function));
```



Étape 2 : les tests

```
it('should toggle all todos to completed', function () {
    // TODO: write test
                /* HB_26_12_2020 doit mettre toutes les taches a completed
                 * créer deux deux taches donc paramètre completed à false
                 * lancer le trigger pour l'événement toggleAll  pour modifier le paramètre completed à true pour les 2 taches
                 * vérifier que le model lise bien les 2 taches en completed false en phase initial
                 * vérifier que le model  met bien à jour les deux taches avec  le paramètre completed à true
                 * test qui échoue vérification du model avec completed false
                 * */
                /*initialisation des deux taches*/
                var todos = [{id:123456,title:'tache1',completed:false},
                    {id:234561,title:'tache2',completed:false}];
                setUpModel(todos);
                subject.setView('');
                view.trigger('toggleAll',{completed:true}
                );
                /* vérification que le model lit bien les taches avec completed à false
                 * test qui échoue avec completed à true
                 */
                expect(model.read).toHaveBeenCalledWith({completed: false}, jasmine.any(Function));
```

Jasmine 2.99.0

30 specs, 0 failures

```
controller
  should show entries on start-up

  routing
    should show all entries without a route
    should show all entries without "all" route
    should show active entries
    should show completed entries

  should show the content block when todos exists
  should hide the content block when no todos exists
  should check the toggle all button, if all todos are completed
  should set the "clear completed" button
  should highlight "All" filter by default
  should highlight "Active" filter when switching to active view

  toggle all
    should toggle all todos to completed
    SPEC HAS NO EXPECTATIONS should update the view
```

Étape 2 : les tests

```
            /*test qui échoue avec completed :false pour les deux ,
             false pour le premier et true pour le deuxième,
             true  pour le premier et false pour le deuxieme
             */
            expect(model.update).toHaveBeenCalledWith(123456,{completed: true}, jasmine.any(Function));
            expect(model.update).toHaveBeenCalledWith(234561,{completed: true}, jasmine.any(Function));
});
```

Jasmine 2.99.0

30 specs, 0 failures

```
controller
  should show entries on start-up

  routing
    should show all entries without a route
    should show all entries without "all" route
    should show active entries
    should show completed entries

  should show the content block when todos exists
  should hide the content block when no todos exists
  should check the toggle all button, if all todos are completed
  should set the "clear completed" button
  should highlight "All" filter by default
  should highlight "Active" filter when switching to active view

  toggle all
    should toggle all todos to completed
    SPEC HAS NO EXPECTATIONS should update the view
```

Étape 2 : les tests

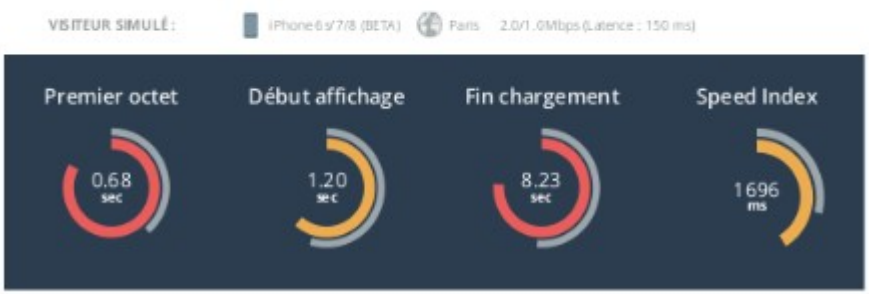
Ajout d'un test
Dans describe toogle All

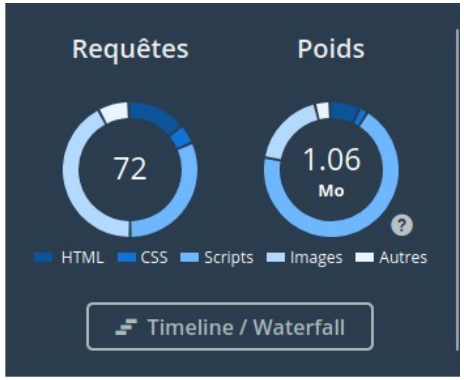
```
it('should toggle all todos to active',function(){
    /*creation de deux taches*/
    var todos = [{id:341256,title:'tache6',completed:true},
                 {id:342156,title:'tache7',completed:true}];
        setUpModel(todos);
        subject.setView('');
       view.trigger('toggleAll',{completed:false});
        /* vérification que le model lit bien les taches avec completed à true
         * test qui échoue avec completed à false*/
        expect(model.read).toHaveBeenCalledWith({completed:true}, jasmine.any(Function));

        /*test qui echoue avec completed :true, false pour le premier et true pour le deuxième,
        true  pour le premier et false pour le deuxieme*/
        expect(model.update).toHaveBeenCalledWith(341256,{completed: false}, jasmine.any(Function));
        expect(model.update).toHaveBeenCalledWith(342156,{completed: false}, jasmine.any(Function));
});
```

```
should show the content block when todos exists
should hide the content block when no todos exists
should check the toggle all button, if all todos are completed
should set the "clear completed" button
should highlight "All" filter by default
should highlight "Active" filter when switching to active view

toggle all
  should toggle all todos to completed
  should update the view
  should toggle all todos to active
```

Étape 3 : optimiser la performance

Audit de performance :
l'application concurrente à auditer est toDoListMe.
L'audit a été réalisé avec l'outil Dareboost qui permet de simuler des visiteurs avec mobile ou en desktop, et les outils de la console web.
Le score obtenu retranscrit le niveau de la page web en tenant compte des enjeux de performance, de sécurité, d'accessibilité ou encore de référencement naturel.





Étape 3 : optimiser la performance

Url	Domain	Time ∨
jquery-ui.js	code.jquery.com	2.1s
show_ads_impl.js	pagead2.googlesyndication.com	1.5s
all.js	connect.facebook.net	1.3s
widgets.js	platform.twitter.com	1.2s
show_ads.js	pagead2.googlesyndication.com	1.1s
javascript_e.js	todolistme.net	926ms

L' analyse de cet audit à été faite par la lecture des graphes et données recueillies concernant notamment les vitesses de chargement selon les types de fichiers .
Ce qui a permis de mettre en avant les faiblesses de l'application concurrente suivantes :

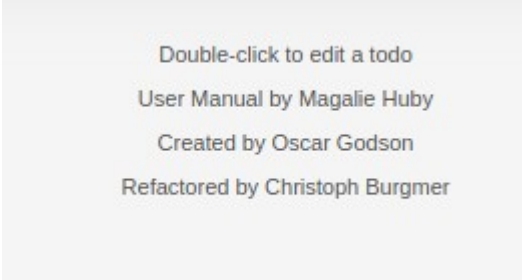
– temps de chargement (scripts, images, documents...)
– site non optimisé pour l'usage des mobiles (qui pourtant représente la plus grande part des accès web à l'heure actuelle)
– un recours trop important à la publicité/réseaux sociaux
– le non respect des standards pour le HTML
– un volume de données très important

Ce qui nous a permis de mettre en avant plusieurs axes d'optimisation pour l'application To-do-list qui sont :
– Mobile First
–Utilisation des fichiers minifiés
– Limitation du nombre d'image et utilisation des sprites
– Optimisation de l'utilisation du cache
– Ajout de nouvelles fonctionnalités à étudier notamment la notion de catégorie et de tri par date

Étape 4 : améliorer le projet

➞ Documentation Utilisateur :

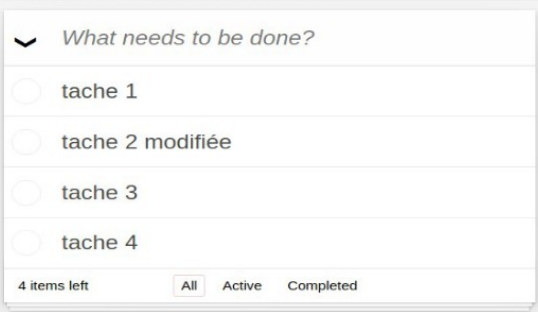
En ce qui concerne la documentation utilisateur elle sera au format PDF et accessible à partir du site par un lien dans le footer avec ouverture dans un nouvel onglet .



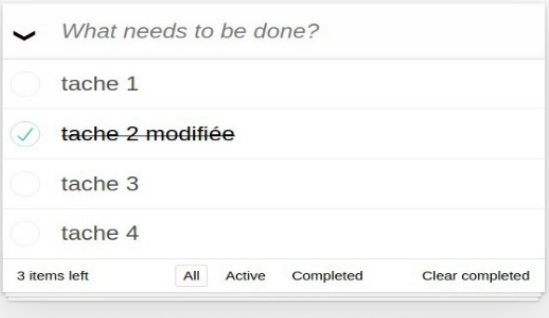
Étape 4 : améliorer le projet

Elle reprend chaque action possible sur l'application et est présentée de sorte que la description de ces actions soit accessible à partir de n'importe quelle page du guide d'utilisation .

- présentation de l'application
- créer une tâche
- modifier le titre d'une tache
- marquer une tache comme terminée
- marquer toutes les taches comme terminées
- visualiser toutes les taches
- visualiser toutes les taches actives
- visualiser toutes les taches terminées
- supprimer une tache
- supprimer toutes les taches à partir de la vue All
- supprimer toutes les taches à partir de la vue Active



Cliquez à gauche de la tache que vous voulez marquer comme terminée pour la séléctionner.
Une icone verte sera affichée et la tache sera barrée.



Le compteur en bas à gauche marque '3 items left'

Étape 4 : améliorer le projet

➞ Documentation technique :

La documentation technique a été générée automatiquement en utilisant documentation.js qui permet d'extraire les blocs de commentaires présent dans le code.
Elle contient également la documentation pour l'ensemble des tests .
Elle se trouve dans le répertoire docs de l'application et est visible avec url /docs/index.html

Étape 4 : améliorer le projet

➜ Audit :

Un audit du site en version initial de l'application todos a été fait .
Un second audit après modification suite aux axes d'amélioration soulevés à l'étape3.
Les comparaisons avant/après et todoListMe/todo ont également été faites .
Les résultats complets sont disponibles dans docs/audit/avant_après.pdf et docs/audit/todo_tolistme.pdf

Étape 4 : améliorer le projet

Comparaison de qualité et performance web

projet8.hubymagalie.loamok.org/

86% 6 4 73

3 janv. 2021 14:09:36 , iPhone 6s/7/8 (BETA) , Paris

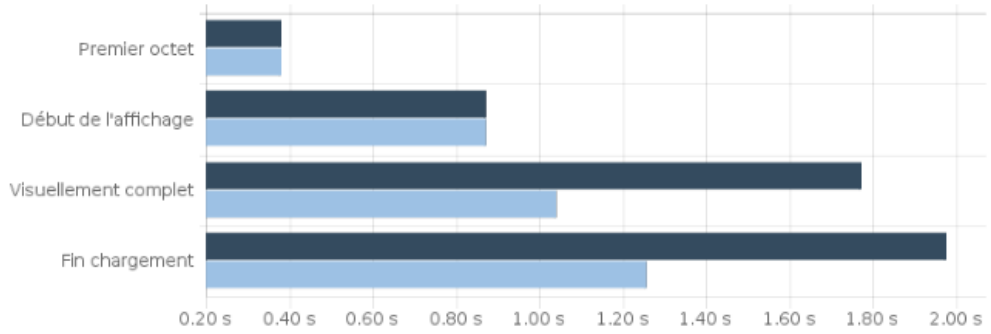
projet8.initial.hubymagalie.loamok.org

72% 9 5 64

3 janv. 2021 14:09:36 , iPhone 6s/7/8 (BETA) , Paris

Légende : projet8.hubymagalie.loamok.org/ projet8.initial.hubymagalie.loamok.org

Temps de chargement



Trafic réseau et requêtes

Nom	Poids total	Total de requêtes	Support HTTP/2
projet8.hubymagalie.loamok.org/	23ko	13	0%
projet8.initial.hubymagalie.loamok.org	20ko	13	0%

Étape 4 : améliorer le projet

Comparaison de qualité et performance web

projet8.hubymagalie.loamok.org/

86% 6 4 73

3 janv. 2021 14:12:25 , iPhone 6s/7/8 (BETA) , Paris

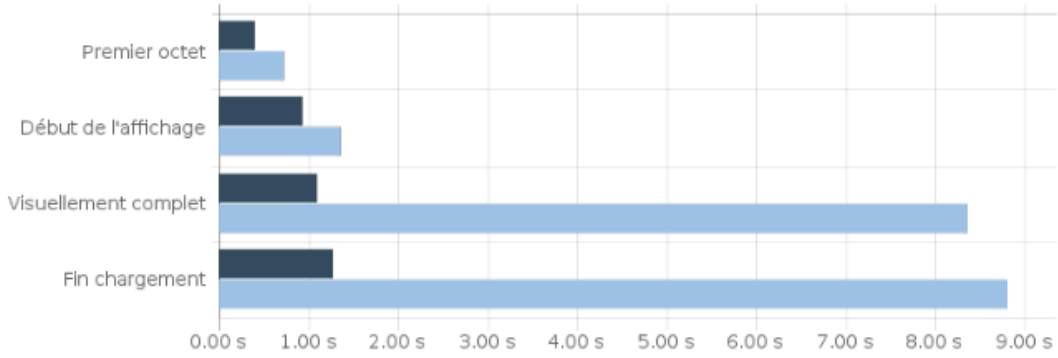
todolistme.net/

59% 14 10 71

3 janv. 2021 14:12:25 , iPhone 6s/7/8 (BETA) , Paris

Légende : projet8.hubymagalie.loamok.org/ todolistme.net/

Temps de chargement



Trafic réseau et requêtes

Nom	Poids total	Total de requêtes	Support HTTP/2
projet8.hubymagalie.loamok.org/	23ko	13	0%
todolistme.net/	1.18Mo	89	48%