

Fiche d'investigation de fonctionnalité

Fonctionnalité : Moteur de recherche

Problématique : Afin de se différencier des concurrents sur le marché, le site *Les petits plats* souhaite mettre en place un moteur de recherche fluide et rapide. Cas d'utilisation en [Annexe 4](#)

Option 1 : Algorithme basé sur la programmation fonctionnelle ([Annexe 1](#))

Cette option utilise les standards contemporains de Javascript (ES5+) basé sur les méthodes de l'objet Array (foreach, filter, map, reduce).

Avantages :

- Lisibilité du code
- Moins de risques d'erreurs accidentelles
- Implémentation de nouveau filtre rapide

Inconvénients :

- Compatibilité avec les anciens navigateurs

Option 2 : Algorithme basé sur des boucles natives ([Annexe 2](#))

Cette option utilise les boucles for et while pour parcourir les données.

Avantages :

- Compatibilité avec les anciens navigateurs

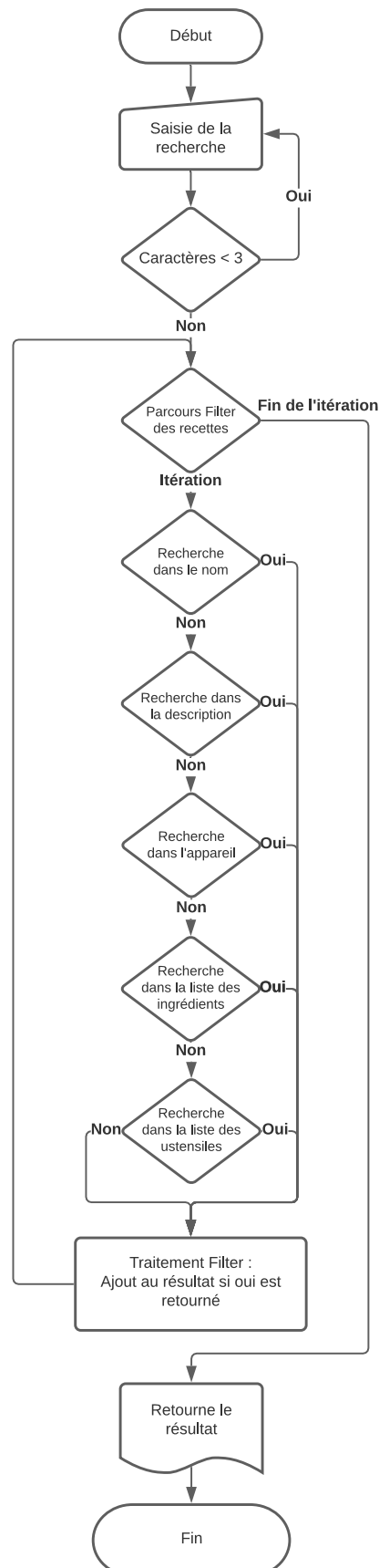
Inconvénients :

- Lisibilité du code
- Risques d'erreurs accidentelles dues à la complexité du code
- Implémentation de nouveau filtre plus lente

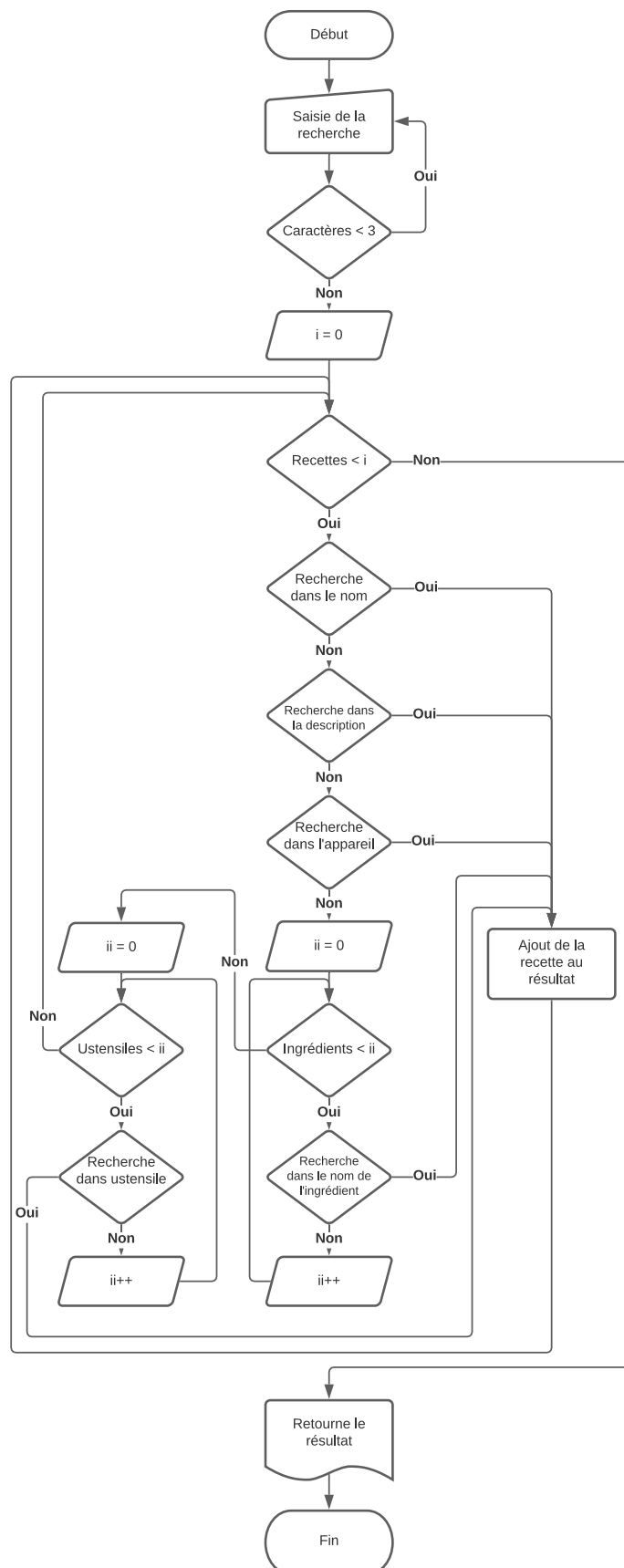
Solution retenue :

Suite aux tests réalisés ([Annexe 3](#)) entre les 2 méthodes sur [JSBench.me](#), l'option 1 a été retenue. Elle s'est démarquée par sa rapidité (entre 16% et 26% plus rapide que l'option 2. Et il est clair qu'elle se démarque également par sa maintenabilité et la facilité d'implémenter de nouvelles options de recherche par le futur.

Annexe 1 - Algorithme option 1



Annexe 2 - Algorithme option 2



Annexe 3 - Résultats des tests

JSBench.Me Run Save

enter test suite name v1 - by -

enter test suite description

+ Setup HTML - click to add setup HTML

+ Setup JS - click to add setup JavaScript

Option 1	<pre>const data = [{ "id": 1, "name": "Limonade de Coco", "servings": 1, "ingredients": [{ "ingredient": "Lait de coco", "quantity": 400, "unit": "ml" }] }]</pre>
finished	
6817.9 ops/s \pm 3.23%	
Fastest	
Option 2	<pre>const data = [{ "id": 1, "name": "Limonade de Coco", "servings": 1, "ingredients": [{ "ingredient": "Lait de coco", "quantity": 400, "unit": "ml" }] }]</pre>
finished	
5650.19 ops/s \pm 3.26%	
17.13 % slower	

+ Test Case - click to add another test case

+ Teardown JS - click to add teardown JavaScript

+ Output (DOM) - click to monitor output (DOM) while test is running

▶ RUN again

[Wiki](#) | [Report issue](#) | [Become a sponsor!](#)

Inspired by [Benchmark.js](#), [Jsperf.com](#) and [jsfiddle.com](#).

Annexe 4 - Cas d'utilisation

