

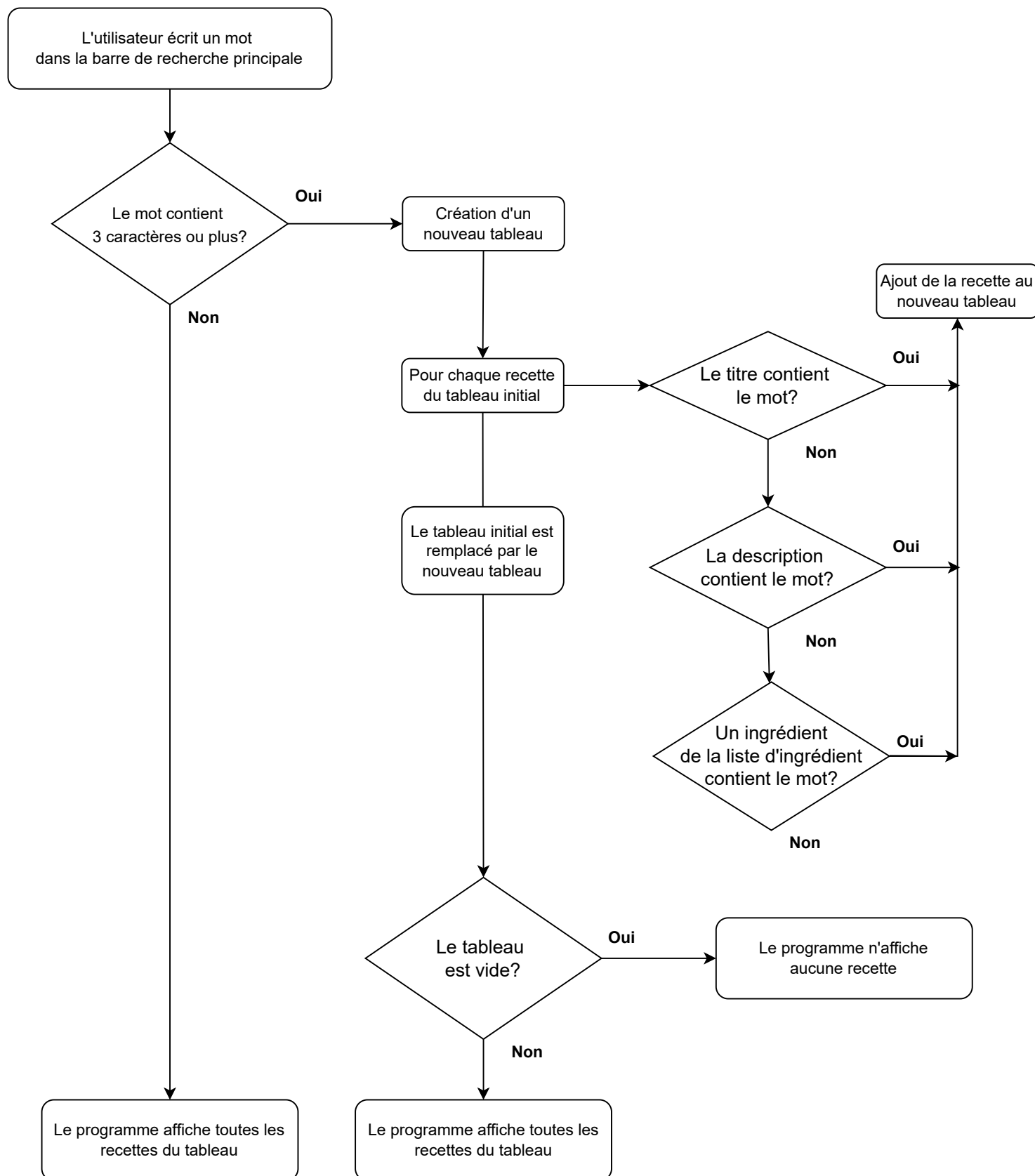
## Fiche d'investigation de fonctionnalité

Fonctionnalité: Recherche principale	Fonctionnalité #2
<p><b>Problématique:</b> Afin d'améliorer l'expérience utilisateur, nous souhaitons mettre en place une fonctionnalité de recherche qui soit la plus rapide possible.</p> <p>Le site devra renvoyer les recettes dont le titre, la description ou les ingrédients contiennent la chaîne de caractères entrée par l'utilisateur.</p> <p>Pour réaliser ceci, l'algorithme analysera le tableau contenant toutes les recettes disponibles, préfiltré ou non par l'utilisation de filtres/tag. Si aucun tag/filtre n'est choisi, le programme analysera le tableau contenant l'ensemble des recettes.</p> <p>L'entrée utilisateur se fera sur le champ de recherche principal en haut de page.</p>	

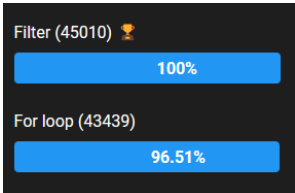
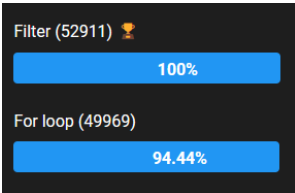
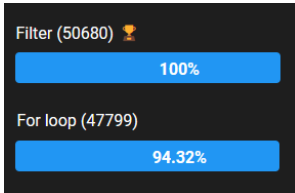
<p><b>OPTION 1: Traitement de tableaux à l'aide des méthodes for/if</b></p> <p>Dans cette option, nous programmons à l'aide de méthodes classiques: boucles for et conditions if.</p> <p>Cela implique de créer un nouveau tableau, une variable pour les ingrédients et d'utiliser deux boucles for / conditions if qui rajoutent la recette dans le nouveau tableau à renvoyer si la recette remplit les conditions de recherches de l'utilisateur</p>	
<b>Avantages</b> <ul style="list-style-type: none"><li>- Code plus facile à aborder</li></ul>	<b>Inconvénients</b> <ul style="list-style-type: none"><li>- Nécessite d'écrire plus de lignes de code</li><li>- Plus lent car nécessite plus de calculs</li><li>- Plus difficile à maintenir</li></ul>

<p><b>OPTION 2: Traitement de tableaux à l'aide des méthodes array filter et some</b></p> <p>Dans cette option, nous programmons à l'aide de méthodes modernes: filter() et some().</p> <p>Cela implique de renvoyer un nouveau tableau qui remplit les conditions de recherches de l'utilisateur. Dans la méthode filter, on utilise la méthode some pour accepter une recette si un de ses ingrédients correspond à la recherche.</p>	
<b>Avantages</b> <ul style="list-style-type: none"><li>- Code plus court et lisible</li><li>- Nécessite moins de calculs donc plus rapide</li><li>- Plus facile à maintenir</li></ul>	<b>Inconvénients</b> <ul style="list-style-type: none"><li>- Les méthodes utilisées sont récentes et peuvent être méconnues (some)</li></ul>

## Algorithme



### Résultats des benchmarks

Ail	Banane	Blender
		
<a href="https://jsben.ch/T7qQD">https://jsben.ch/T7qQD</a>	<a href="https://jsben.ch/y9RPS">https://jsben.ch/y9RPS</a>	<a href="https://jsben.ch/TKOaj">https://jsben.ch/TKOaj</a>

### Conclusion

L'algorithme de recherche avec les méthodes modernes `filter()` et `some()` est plus performant et beaucoup moins verbeux avec une diminution de texte d'environ 50%.

Il est aussi plus maintenable parce qu'en cas de changement sur la structure des données, les méthodes utilisées sont plus rapides à modifier que les boucles `for`.

L'algorithme moderne avec les méthodes `filter` et `some` sera utilisé pour le site Les petits plats

### Liens annexes

**Github:** [https://github.com/Nabil-Y/NabilYassine\\_7\\_13032022/tree/dev](https://github.com/Nabil-Y/NabilYassine_7_13032022/tree/dev)

**Branche algorithme 1:** [https://github.com/Nabil-Y/NabilYassine\\_7\\_13032022/tree/searchFor](https://github.com/Nabil-Y/NabilYassine_7_13032022/tree/searchFor)

**Fonction recherche algorithme 1:**

[https://github.com/Nabil-Y/NabilYassine\\_7\\_13032022/blob/searchFor/scripts/search.js](https://github.com/Nabil-Y/NabilYassine_7_13032022/blob/searchFor/scripts/search.js)

**Branche algorithme 2:** [https://github.com/Nabil-Y/NabilYassine\\_7\\_13032022/tree/searchMap](https://github.com/Nabil-Y/NabilYassine_7_13032022/tree/searchMap)

**Fonction recherche algorithme 2:**

[https://github.com/Nabil-Y/NabilYassine\\_7\\_13032022/blob/searchMap/scripts/search.js](https://github.com/Nabil-Y/NabilYassine_7_13032022/blob/searchMap/scripts/search.js)