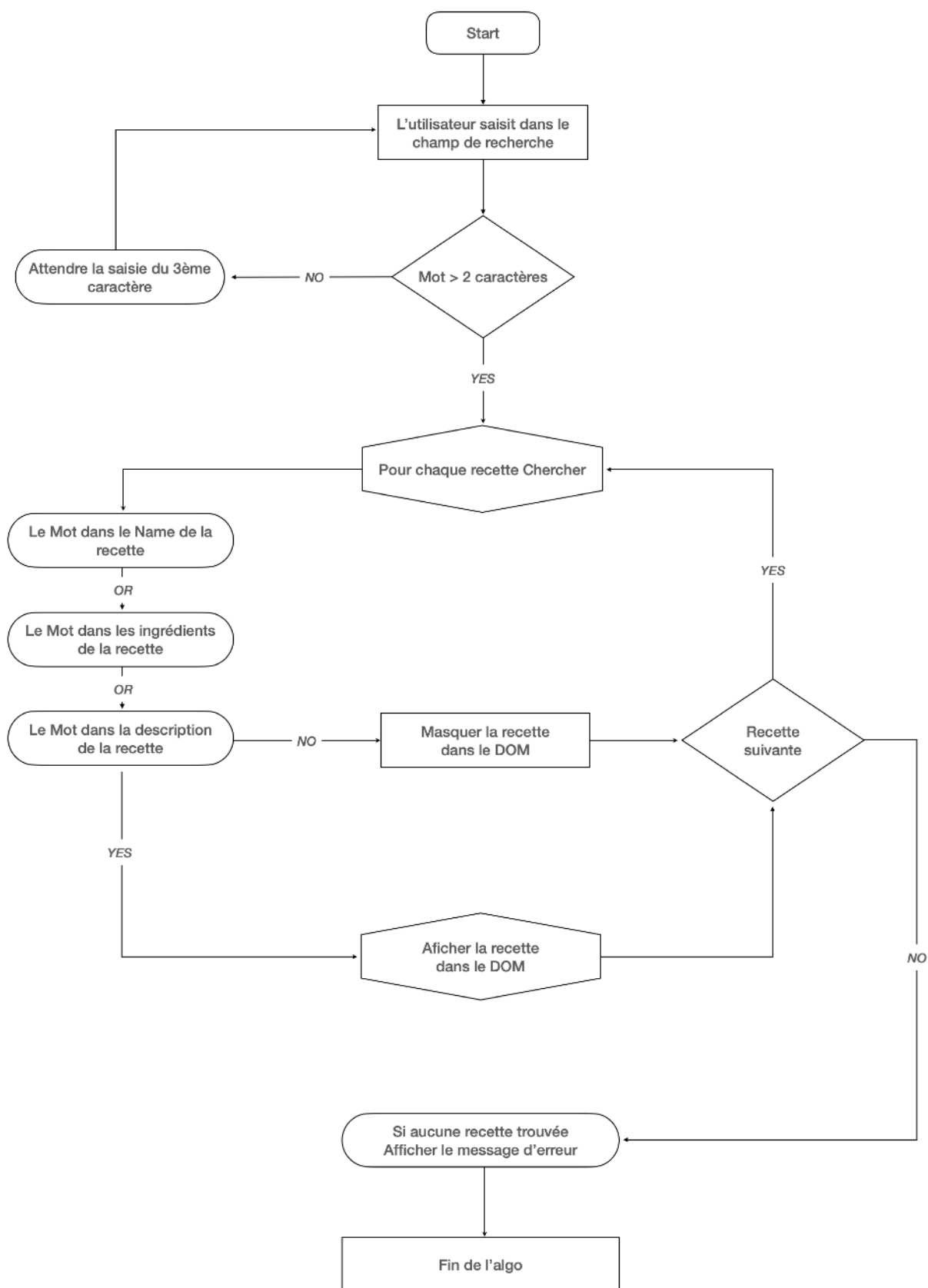


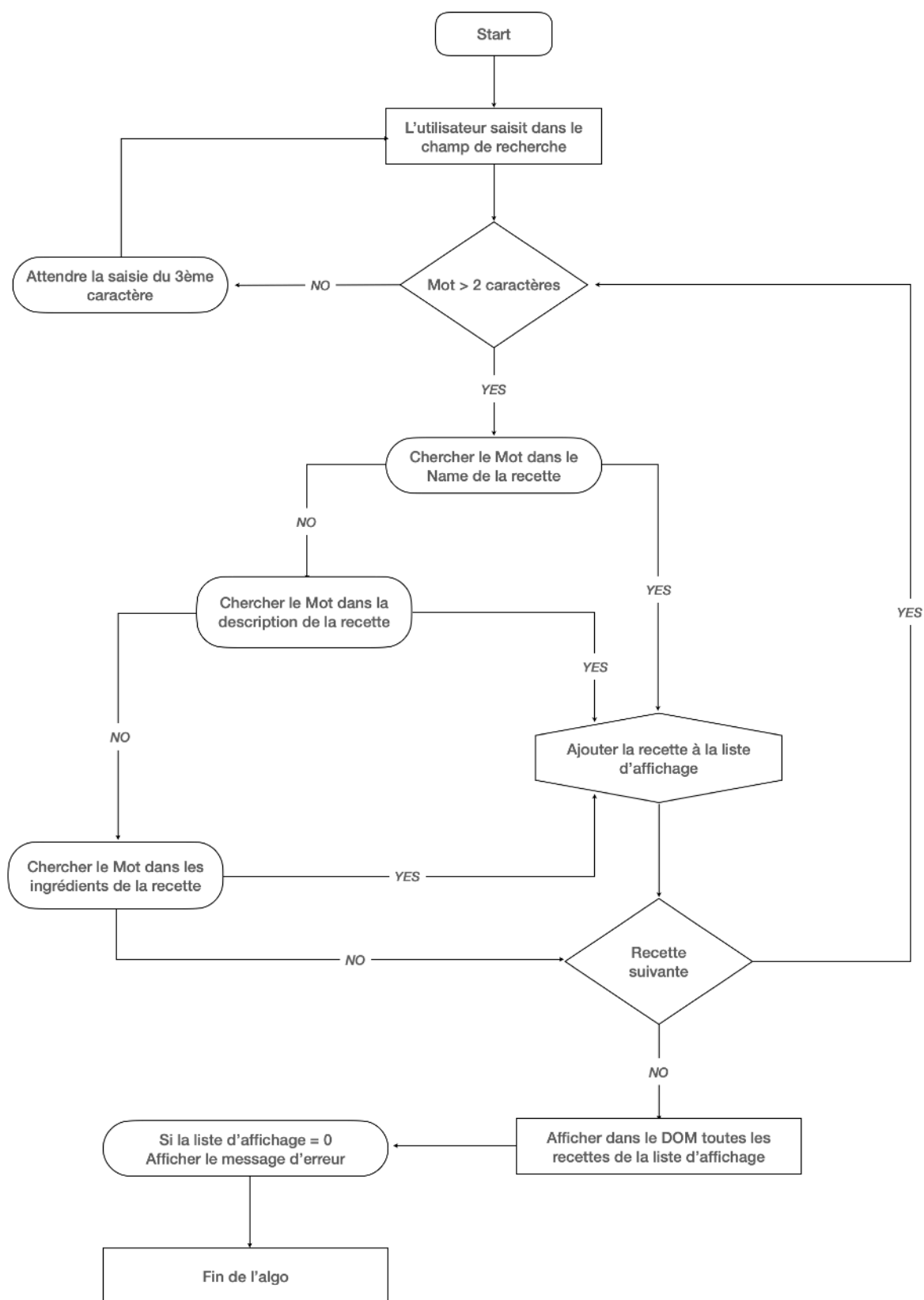
## Fiche d'investigation de fonctionnalité

<b>Fonctionnalité</b> : Recherche de recettes	
<p><b>Problématique</b> : Réaliser une recherche rapide et performante dans les titres, les descriptions et les ingrédients des recettes. L'utilisateur doit pouvoir filtrer les recettes selon deux axes : une barre principale pour rechercher des mots ou des groupes de lettres dans le titre, les ingrédients ou la description et une recherche par mots-clés dans les ingrédients, les ustensiles ou l'appareil. L'analyse suivante concerne la recherche de la barre principale, fournissant 2 algorithmes et une proposition finale pour le plus efficace.</p>	
<p><b>Option 1 : MainSearchV1 en programmation fonctionnelle avec filter()</b></p> <p>Dans cette option, l'algorithme prend le groupe de lettres saisi par l'utilisateur dans la barre de recherche principale, s'elles dépassent trois lettres l'algo filtre les recettes en conséquence. Ce code exécute la fonction de recherche dans le nom de la recette ou sa description ou ses ingrédients. S'il ne trouve aucune correspondance il masque la recette dans le DOM. Cette opération est répétée pour toutes les recettes, et si à la fin aucune recette n'est trouvée alors il affiche le message « Aucune recette ne correspond à votre recherche... ».</p>	
<p><b>Avantages</b></p> <ul style="list-style-type: none"> <li>- Facile à utiliser et faire évoluer au cas où d'autres paramètres de recherche seraient ajoutés;</li> <li>- ressources mémoire réduite.</li> </ul>	<p><b>Inconvénients</b></p> <ul style="list-style-type: none"> <li>- Plus d'opérations ce qui peut rendre le processus de filtrage plus lent.</li> </ul>
<p><b>Nombre de champs minimum pour lancer la recherche : 3</b></p> <p>Recherche dans le champ du nom, des ingrédients et de la description</p>	
<p><b>Option 2 : mainSearchV2 en algorithme de tri avec boucle for</b></p> <p>Dans cette option, l'algo prend le groupe de lettres saisis s'ils dépassent les trois caractères, et effectue la recherche dans le nom de la recette, s'il trouve une correspondance il ajoute la recette à la liste d'affichage sinon il recherche dans la description, s'il trouve une correspondance il ajoute la recette à la liste d'affichage sinon il fait la même opération avec les ingrédients. Après avoir fait la totalité des recettes, il affiche dans le DOM les recettes figurants dans la liste d'affichage, si cette liste est vide il affiche le message « Aucune recette ne correspond à votre recherche... »</p>	
<p><b>Avantages</b></p> <ul style="list-style-type: none"> <li>- L'algorithme de tri rend la recherche plus rapide à exécuter.</li> </ul>	<p><b>Inconvénients</b></p> <ul style="list-style-type: none"> <li>- Plus dur à comprendre en raison de la structure plus complexe de l'algorithme.</li> <li>- Le code n'est pas très flexible et peut être difficile à maintenir si vous voulez ajouter de nouveaux éléments de recherche.</li> </ul>
<p><b>Nombre de champs minimum pour lancer la recherche : 3</b></p> <p>Recherche dans le champ du nom, des ingrédients et de la description</p>	
<p><b>Solution retenue :</b></p> <p>Je choisis d'utiliser la fonction mainSearchV1 car bien que les résultats de performance soient légèrement inférieurs à l'option 2, la différence est si minime (voir resultat JSBen.ch) qu'elle n'a aucun impact significatif pour 50 recettes et est imperceptible pour les utilisateurs. Cependant, à grande échelle, il serait préférable d'utiliser mainSearchV2.</p> <p>En outre, le code de mainSearchV1 est plus facile à comprendre et plus lisible car il n'y a aucun algorithme complexe visible dans la fonction, ce qui la rend plus simple et plus maintenable.</p>	

**Approche 1**




**Approche 2**




## Résultats de la comparaison d'algorithmes avec jsben.ch

<https://jsben.ch/dlmsg>

JSBEN.CH

mainSearchV1 

```
1 function mainSearchV1(e) {
2   let searchedWords = [];
3   if (e.target.value === '') {
4     searchedWords = [...allSelectedTags].map((tag) =>
5       tag.toLowerCase().replace(/ /g, ''))
6   };
7 } else {
8   searchedWords = [
9     ...e.target.value.toLowerCase().split(' '),
10    ...allSelectedTags.map((tag) => tag.toLowerCase().replace(/ /g, '')),
11  ];
12 }
```

code block 2 

```
1 function mainSearchV2(e) {
2   let searchedWords = [];
3   if (e.target.value === '') {
4     searchedWords = [...allSelectedTags].map((tag) =>
5       tag.toLowerCase().replace(/ /g, ''));
6   } else {
7     searchedWords = [
8       ...e.target.value.toLowerCase().split(' '),
9       ...allSelectedTags.map((tag) => tag.toLowerCase().replace(/ /g, '')),
10    ];
11  }
12 }
```

