# Fiche d'investigation de fonctionnalité

| **Fonctionnalité** : Filtrer les recettes | **Fonctionnalité** #3 |
|---|---|
| **Problématique** :<br>Afin d'accéder rapidement à une recette, nous cherchons à trouver le meilleur algorithme de tri pour filtrer les recettes.<br>Quel algorithme sera le plus performant pour réaliser une recherche sur les noms, descriptions et ingrédients d'un tableau de recettes ? | |

| **Option 1 : Algorithme avec des boucles natives.**<br>Dans cette option, nous choisissons d'utiliser des boucles natives.<br>Nous effectuons la recherche en itérant chaque recette à l'aide d'une boucle For et nous cherchons les correspondances avec le nom ou la description et sinon dans les ingrédients.<br>Si une recette correspond nous l'insérons dans un nouveau tableau. *Voir annexe page 3* | |
|---|---|
| **Avantages :**<br>- Les boucles natives sont plus rapides pour parcourir un tableau.<br>- Elles permettent de répéter des actions simplement.<br>- Elles sont compatibles avec tous les navigateurs.<br>- Elles sont très modulables et facilement utilisables. | **Inconvénients :**<br>- Elles deviennent plus lente lorsqu'elles sont imbriquées. |

| **Option 2 : Algorithme avec de la programmation fonctionnelle**<br>Dans cette option, nous choisissons d'utiliser de la programmation fonctionnelle.<br>Nous filtrons le tableau des recettes avec la méthode filter() et avec une fonction callback nous cherchons les correspondances dans le nom, la description et les ingrédients que nous avons préalablement mis dans un tableau avec la méthode map().<br>La méthode filter() permet de retourner un tableau avec tous les éléments correspondants.<br>*Voir annexe page 4* | |
|---|---|
| **Avantages :**<br>- La programmation fonctionnelle permet d'avoir un code plus compréhensible et plus court.<br>- Les méthodes filter() et map() sont compatibles sur tous les navigateurs.<br>- Elles sont spécifiques à une certaines utilisations et dans ce cas très rapide. | **Inconvénients :**<br>- Leur application est plus difficile.<br>- Elles ne sont pas modulables |

| **Solution retenue :**<br>La solution retenue est celle de l'option 2.<br><br>A l'aide d'un outil de comparaison de performance (JSBENCH), nous avons pu analyser les deux algorithmes. Nous avons pu constater que pour 50, 100 ou 1000 recettes, qu'on recherche un terme « coco » ou un ingrédient « lait de coco » et à partir du navigateur Chrome ou Firefox. *Voir annexe page 2* |
|---|

# Annexes

## Tableau de comparaison

| Test | | | Nb d’opération/sec | | Différence en % |
|---|---|---|---|---|---|
| recherche | Nb de recette | navigateur | Algo 1 | Algo 2 | |
| Coco | 50 | Chrome | 51852 | 59361 | 12.65% |
| Coco | 50 | Firefox | 37473 | 46151 | 18.8% |
| Lait de coco | 50 | Chrome | 48982 | 56630 | 13.51% |
| Coco | 100 | Chrome | 27334 | 31457 | 13.11% |
| Coco | 1000 | Chrome | 2806 | 3240 | 13.4% |
| Lait de coco | 1000 | Chrome | 2219 | 2850 | 22.14% |

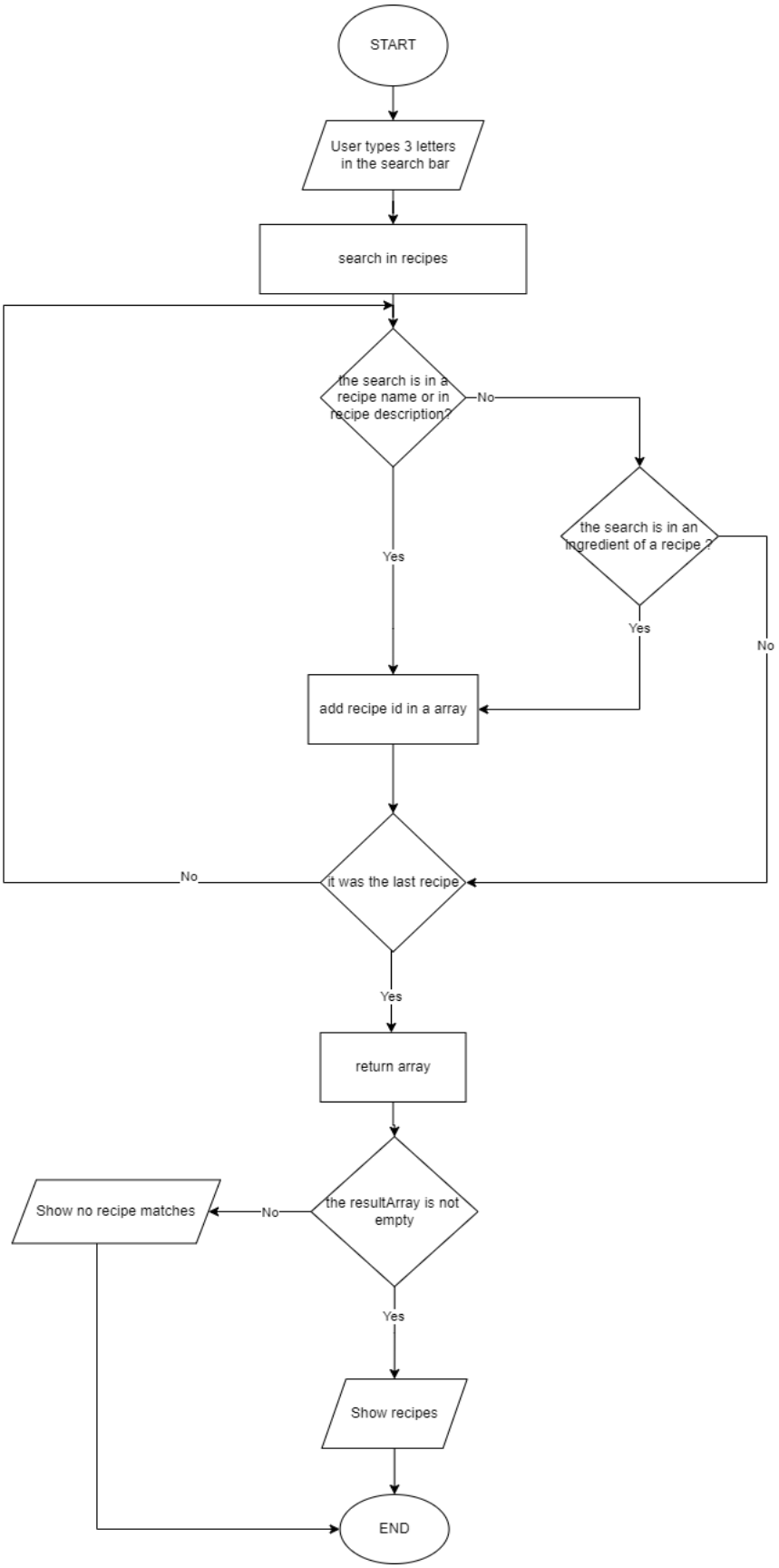


**Figure 1 : Diagramme avec boucles natives.**

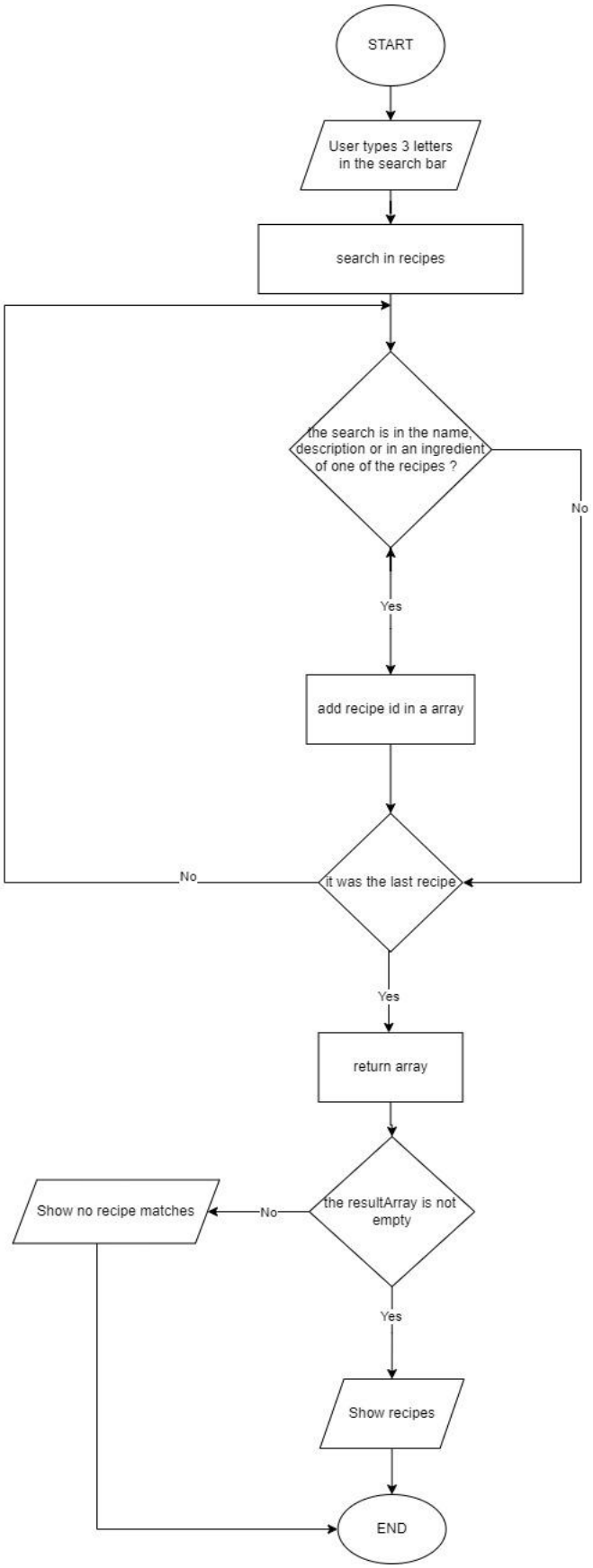


**Figure 2 : Diagramme avec programmation fonctionnelle.**