Fiche d'investigation de fonctionnalité

Fonctionnalité : Tri de recette selon ingrédient/ustensils/appareils	**Fonctionnalité #2**
Problématique : Réaliser son propre site de recette de cuisine à l'instar de Marmiton, avec une fonctionnalité de recherche/tri de recette fluide et rapide.	

Option 1 : Algorithme avec des boucles natives (cf Figure)
Dans cette option, nous utiliserons des boucles natives, tels que while et for afin de construire l'algorithme de recherche.

Avantages	**Inconvénients**
⊕ **Contrôle Fin :** Les boucles comme for ou while offrent un contrôle plus fin sur le processus de recherche, y compris la possibilité de sortir de la boucle dès que le critère est satisfait. ⊕ **Performance :** Dans certains cas, les boucles natives peuvent être légèrement plus performantes, car elles évitent l'overhead lié aux appels de méthode et peuvent être optimisées pour des scénarios spécifiques. ⊕ **Flexibilité :** Elles permettent une personnalisation plus poussée et une manipulation directe des indices, ce qui peut être utile pour des algorithmes complexes.	⊖ **Complexité du Code :** L'utilisation de boucles peut rendre le code plus verbeux et moins lisible, surtout pour des opérations de recherche simples. ⊖ **Risques d'Erreurs :** Il est plus facile de faire des erreurs, comme oublier de mettre à jour l'indice ou oublier de sortir de la boucle.

Nombre de champs minimum à remplir à la recherche : 1

Option 2 : Algorithme avec la méthode d'objet des tableaux
Dans cette option, nous utiliserons des boucles avec la méthode d'objet des tableaux, comme 'find' par exemple, afin de construire l'algorithme de recherche

Avantages	**Inconvénients**
⊕ **Lisibilité et Concision :** Les méthodes comme find, some, indexOf, etc., rendent le code plus lisible et plus concis, en réduisant la quantité de code nécessaire pour effectuer des recherches. ⊕ **Expressivité :** Ces méthodes expriment de manière plus claire l'intention de recherche, rendant le code plus facile à comprendre et à maintenir. ⊕ **Sécurité :** Les méthodes intégrées gèrent souvent les cas particuliers et les erreurs, ce qui réduit le risque de bugs.	⊖ **Performance :** Bien que généralement optimisées, ces méthodes peuvent introduire un léger overhead en raison de la gestion des appels de méthode et des abstractions supplémentaires. ⊖ **Moins de Contrôle :** Certaines méthodes n'offrent pas la flexibilité nécessaire pour des opérations de recherche très spécifiques ou complexes. ⊖ **Surcoût Mémoire :** L'utilisation des méthodes d'objet peut parfois entraîner un surcoût mémoire en raison des abstractions et des opérations supplémentaires qu'elles effectuent.

Nombre de champs minimum à remplir à la recherche : 1

Solution retenue :
D'après le résultat jsbench, nous pouvons constater que la méthode avec les boucles natives est légèrement plus performante que la méthode des tableaux. Donc d'un point de vue performance il serait plus intéressant de retenir cette solution. Cependant la différence est assez minime avec la méthode des tableau, et cette dernière présente un avantages pour nous, la maintenance du code sera plus simple avec celle-ci qu'avec les boucles natives. A en discuter pour décider le solution que vous préfèrerez.

Annexes

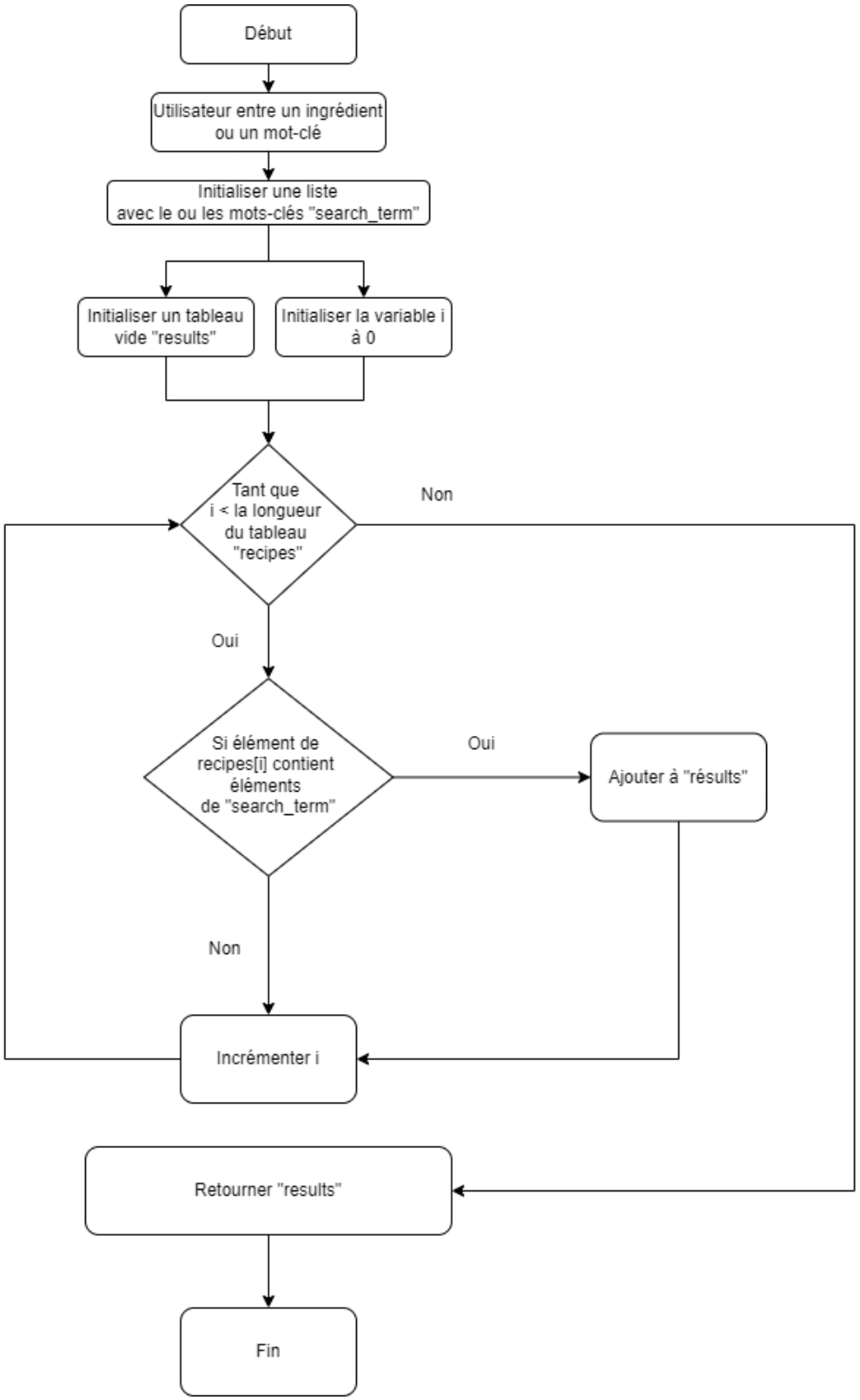

Figure 1 - Diagramme d'algo à boucles natives

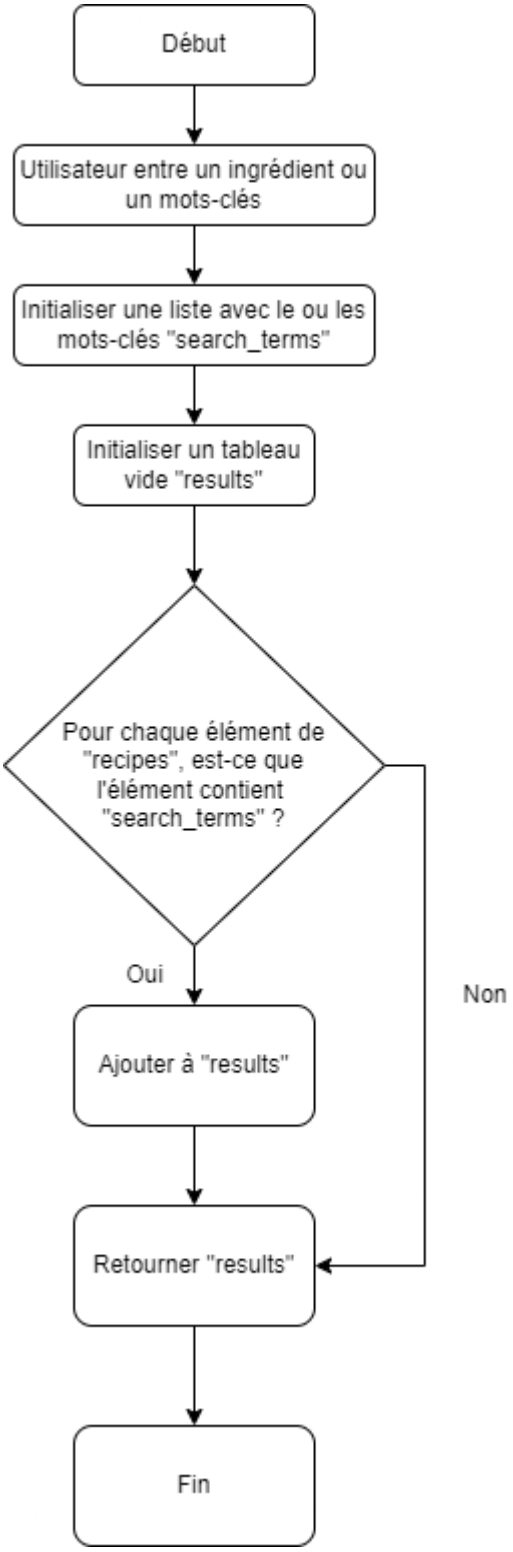

Figure 2 : Diagramme algo méthode d'objet des tableaux