Fiche d’investigation de fonctionnalité

Fonctionnalité : Recherche de recettes + filtres	**Fonctionnalité #1**
Problématique : L’utilisateur doit pouvoir filtrer les recettes selon deux axes : 1- Une barre principale permettant de rechercher des mots ou groupes de lettres dans le titre, les ingrédients ou la description. 2 – Recherche par mots clés dans les ingrédients, les ustensiles ou les appareils.	

Option 1 : Algorithme de recherche se basant sur les boucles natives (while, for…).
Une boucle for répète des instructions jusqu’à ce qu’une condition donnée ne soit plus vériée et ajoute les recettes filtrées dans un tableau avec .push().

Résultat des tests de performance : 1,6 Md ops/s ± 1,6 % (indiquant qu'il s'agit du plus rapide des deux).

Avantages :	***Inconvénients :***
- Rapidité et performance -	- L’algorithme utilisant les boucles natives est plus long à écrire et moins lisible. - Lignes de code plus denses - Présence d’index et d’incrémentation.

Option 2 : Algorithme de recherche se basant sur les méthodes de l’objet array (filter, some, map, reduce).
Cet algorithme utilise les méthodes fonctionnelles comme .filter() et .some() pour filtrer les recettes.

Résultat des tests de performance : 37k ops/s ± 1,29 %, ce qui le rend 100 % plus lent que l'algorithme 1.

Avantages :	***Inconvénients :***
- Pratique car elle est disponible par défaut. - Concise car elle a moins de ligne de code.	- L’appel d’une fonction callback sur chaque élément ralentit la performance sur un tableau de taille conséquente.

Solution retenue :
L’option 1 car elle est plus performante d’après le benchmark Javascript. Tout en gardant la logique même du filtrage malgré la lisibilité et la longueur du code.