**Choix de l’algorithme : Search and Tag**

**1 : Algorythme 1 Search and Tag :**

**Une image contenant diagramme, texte, Plan, Dessin technique

Description générée automatiquement**

**I Effet** : Filtre les recettes en utilisant la méthode filter de JavaScript, qui est chaînée avec d'autres méthodes comme includes et some.

**II Fonctionnement :**

Vérifie si le terme de recherche correspond à la recette (nom, description, ou ingrédients).

Vérifie la présence de tous les tags d'ingrédients dans la recette.

Vérifie la correspondance du tag d'appareil avec l'appareil de la recette.

Vérifie la présence de tous les tags d'ustensiles dans la recette.

**III Efficacité :** Cette méthode est généralement plus concise et peut être plus lisible. Elle utilise des fonctions intégrées de haut niveau qui sont optimisées et peut être plus rapide sur les petits ensembles de données. Cependant, pour les gros ensembles de données, l'utilisation répétée de some et every peut être coûteuse en performance car chaque recette est vérifiée à plusieurs reprises pour chaque tag.

**2 Algorythme 2 Boucle**

Une image contenant texte, diagramme, capture d’écran, Plan

Description générée automatiquement

**I Effet :** Filtre les recettes en utilisant une boucle for classique et des structures de contrôle conditionnelles.

**II Fonctionnement :**

Passe en revue chaque recette individuellement.

Utilise une boucle interne pour vérifier la correspondance du terme de recherche dans les ingrédients si nécessaire.

Vérifie chaque tag d'ingrédient, d'appareil et d'ustensile pour chaque recette.

**III Efficacité** : Cette méthode offre un contrôle plus fin sur le processus de filtrage et peut être plus efficace si le terme de recherche est couramment recherche

**Pourquoi avoir choisi l’algorythme Search and tag ?**

**1 test de vitesse :**

**Formule pour tester 100 fois dans console chrome :**

let nombreIterations = 100;

let tempsTotal = 0;

for (let i = 0; i < nombreIterations; i++) {

    let t0 = performance.now();

    // Simuler une saisie dans le champ de texte

    let input = document.querySelector('input[type="text"]'); // Utilisez le sélecteur approprié pour votre champ de saisie

    input.value = "hou"; // Définissez la valeur de saisie pour la recherche

    input.dispatchEvent(new Event('input')); // Déclencher l'événement input pour simuler la saisie utilisateur

    let t1 = performance.now();

    tempsTotal += t1 - t0;

    console.log(`Test ${i}: ${t1 - t0} ms`);

}

console.log(`Temps d'exécution moyen : ${tempsTotal / nombreIterations} ms`);

**1Algorythme Filtre**

**Le temps d’execution moyen est toujours entre 0.8 ms 1 ms**

**Test 100 fois :**

**Une image contenant texte, capture d’écran, Page web, logiciel

Description générée automatiquement**

**2 Algorythme Boucle**

**Test sur 100 fois avec chrome temps moyen sur 100 fois 1 a 1.2:**

**Une image contenant texte, capture d’écran, Page web, logiciel

Description générée automatiquement**

**Même si les 2 font très bien le travail la fonction Filtre un est plus en adéquation avec le codage moderne et performe plus pour le green code de plus elle est légèrement plus rapide.**

**Cependant la fonction Boucle sera peut etre plus precise avec plus de recettes il sera intéressant de retester notamment la vitesse du site avec la totalité des recettes.**

**Aujourd’hui la fonction filtre remporte le test et de performance.**