Fiche d’investigation de fonctionnalité #2

Fonctionnalité: Recherche de recettes

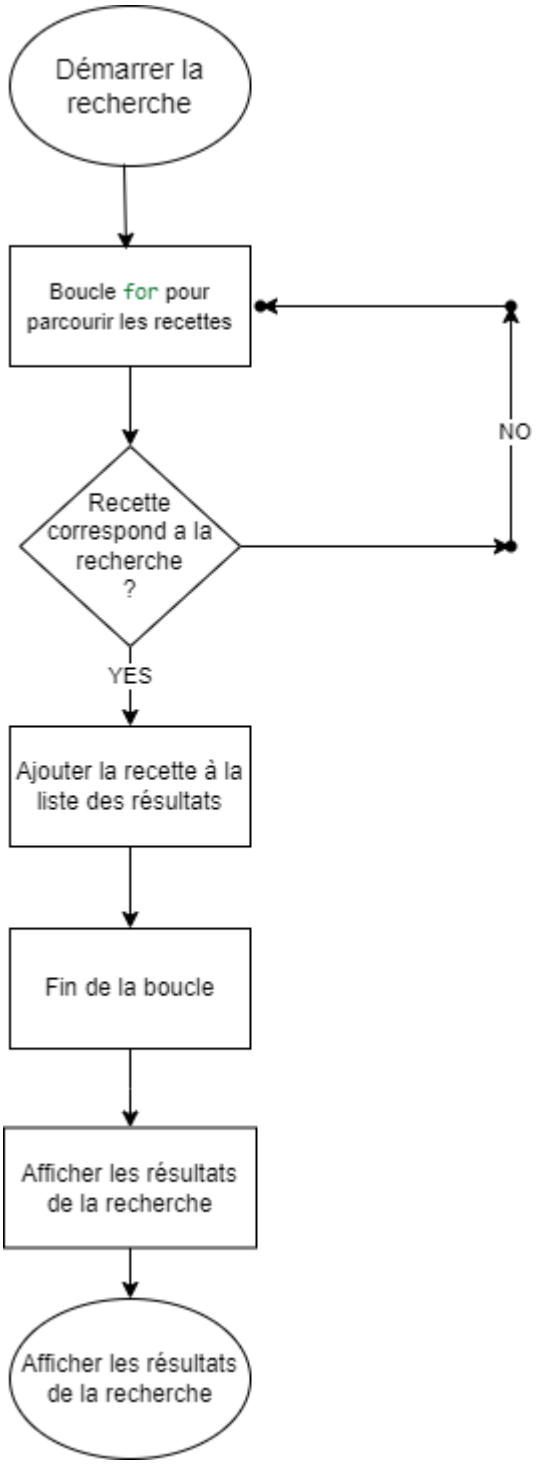
Problématique: Afin de se démarquer sur un marché très concurrentiel, nous avons besoin d’une fonctionnalité de recherche rapide et fluide pour que les utilisateurs puissent trouver instantanément les recettes. Deux approches seront testées pour déterminer la plus performante.

Option 1: Recherche basée sur des boucles natives

Cette option repose sur l'utilisation des boucles classiques de JavaScript (comme `for`, `while`, etc.) pour parcourir et filtrer les résultats du tableau JSON de recettes.

- **Avantages** :
  - Contrôle total sur le flux d'exécution.
  - Facilité à comprendre et à déboguer.
  - Bonne performance pour un volume limité de données.
- **Inconvénients** :
  - Peut devenir inefficace avec un grand nombre de recettes.
  - Nécessite plus de code pour manipuler les données.

Schéma



Voici le diagramme d'activité montrant comment les boucles traversent chaque recette, vérifient les conditions, et ajoutent les recettes correspondantes à une liste de résultats.

Flux avec des étapes claires sur l'itération et la condition pour chaque recette.

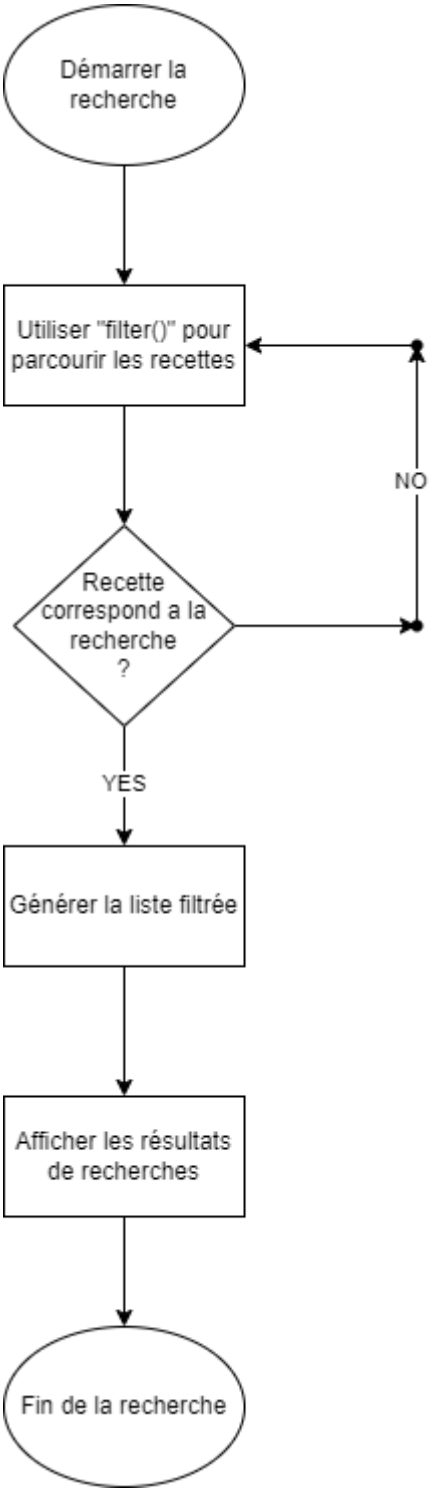
(généré dans draw.io)

Option 2: Recherche fonctionnelle avec les méthodes d’array

Cette option repose sur les méthodes fonctionnelles de JavaScript comme `filter`, `map`, et `reduce`, pour parcourir et filtrer les résultats du tableau JSON.

- **Avantages** :
  - Code plus concis et lisible.
  - Méthodes optimisées en interne par le moteur JavaScript.
  - Permet de traiter les données de manière plus fonctionnelle.
- **Inconvénients** :
  - Peut être plus difficile à comprendre pour un développeur non initié.
  - Moins de contrôle direct sur le flux d'exécution.

Schéma



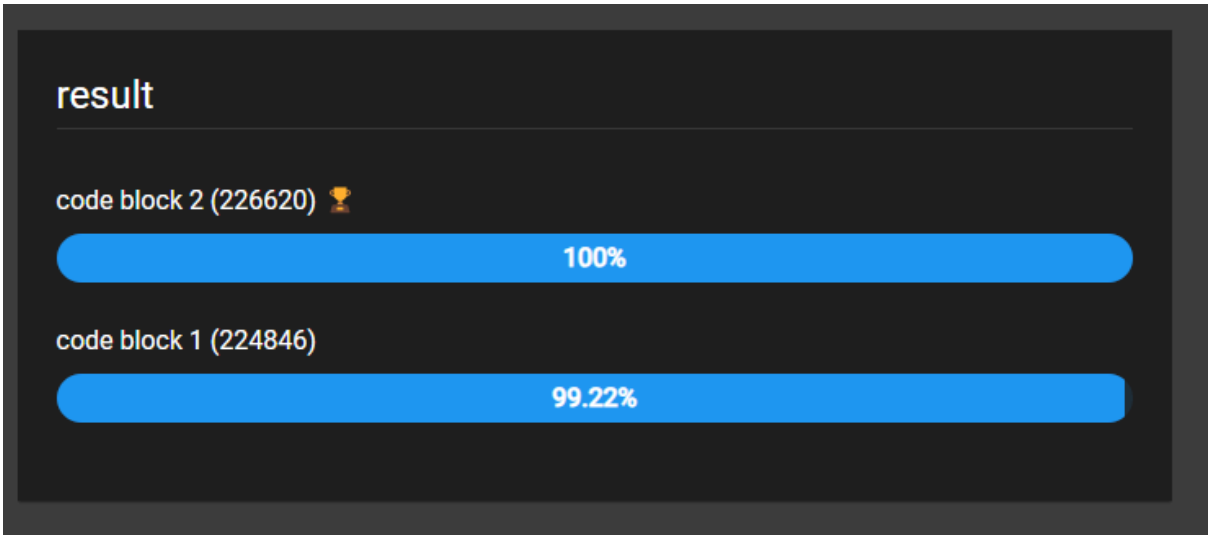
Voici le diagramme d'activité montrant comment les méthodes fonctionnelles traitent le tableau, filtrent les recettes, et renvoient les résultats.

Un diagramme plus simple, mais qui montre l'utilisation de `filter()` et la génération de la liste finale.

(généré dans draw.io)

Résultats des tests de performance avec Jsben.ch.

Ici, nous avons comparé le nombre d'opérations par seconde.
Les résultats déterminent quelle approche est la plus performante en conditions réelles.
URL : https://jsben.ch/tQEqe



(Capture d'écran des résultats des tests de performance.)

Solution retenue :

Une fois les deux approches testées, la **méthode fonctionnelle** semble être la plus appropriée pour un large volume de données, grâce à sa lisibilité et à ses performances optimisées.

La version basée sur des **boucles natives** pourrait cependant être préférée pour des cas plus spécifiques ou nécessitant plus de contrôle.

Recommandation :

Après analyse des performances, je recommande l'algorithme le plus rapide et adapté au volume de données prévu pour la recherche.