Fiche d’investigation de fonctionnalité

Fonctionnalité: recherche / recherche avancée	**Fonctionnalité #1**

Problématique:

Accéder rapidement à une recette correspondant à un besoin de l’utilisateur dans les recettes déjà reçues. Pour tester la rapidité de la fonction de recherche, nous retenons 2 options pour la construire.

- option 1: utiliser les méthodes natives de JS.
- option 2: utiliser les méthodes des arrays

Description :

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: utilisation des méthodes natives de JS pour parcourir et filtrer les recettes

Dans cette option, nous utilisons les méthodes natives (boucles for) de Javascript pour parcourir les tableaux de recettes et garder celles qui correspondent aux critères de l’utilisateur.

Avantages:	**Inconvénients:**
- plus rapide à exécuter - contrôle de la boucle plus précis (départ, fin, incrémentation) - Pas besoin d’instancier un objet pour l'utiliser	- code plus complexe à écrire => risque plus importants de bug - besoin de préciser élément modifié

option 2: utilisation des méthodes spécifiques aux arrays pour parcourir et filtrer les recettes

Dans cette option, nous utilisons les méthodes spécifiques aux arrays (filter, foreach) pour parcourir les tableaux de recettes et garder celles qui correspondent aux critères de l’utilisateur.

Avantages:	**Inconvénients:**
- plus facile à écrire /lire - plus facile à utiliser (retourne un élément à chaque fois)	- moins rapide lors de l’exécution - besoin d’instancier un tableau pour appeler méthode - itération non modulable (lie un tableau du début à la fin)

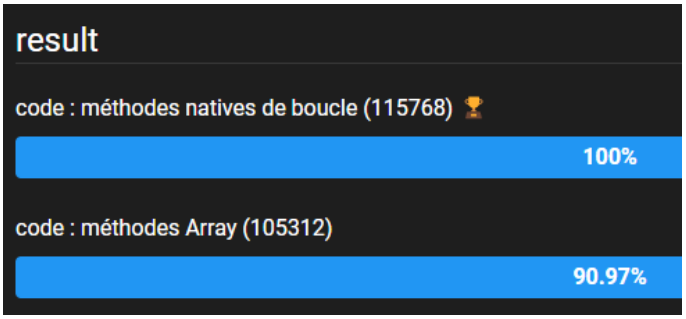
Test de performances:

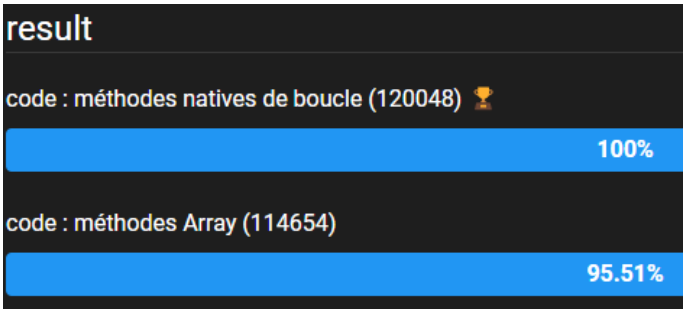
Les tests ont été effectués sur JSBEN.CH avec un tableau contenant 50 recettes.
Le mot-clé “coco” a été utilisé pour la partie recherche global.
Dans un premier temps, les tests ont été effectués sans tag puis avec une recherche multi-tags (“poulet”, “tomate”, “saladier”, et “presse citron”).
Les runs ont été lancés 10 fois sur chaque test.
titre du test dans JSBEN.CH => “Test performance fonction de recherche boucles For vs forEach + filter”

Des tests ont aussi été effectués pour comparer la rapidité des boucles for vs les méthodes map et array sur un plus grand nombre d’itérations (plus de 800).
hypothèses: action push simplement tester sous condition.
 => titre du test sur JSBEN.CH “test rapidité boucle for vs map vs foreach avec multiples conditions et un grand nb d'itérations”

Résultat des tests sur une base 50 recettes sans tag:

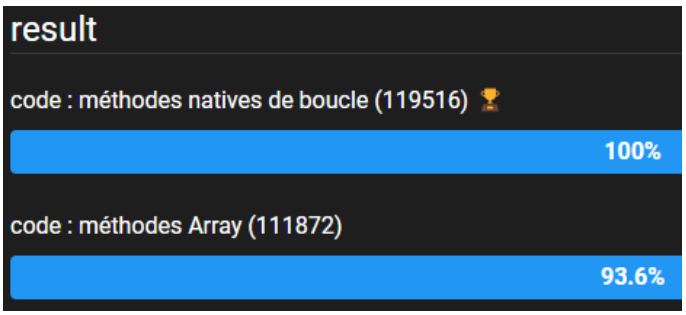
Sur une dizaine de run, l’option 1 avec les boucles natives a été plus rapide à chaque fois (100%)

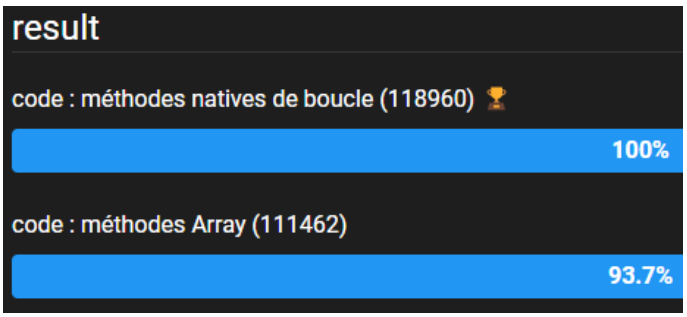




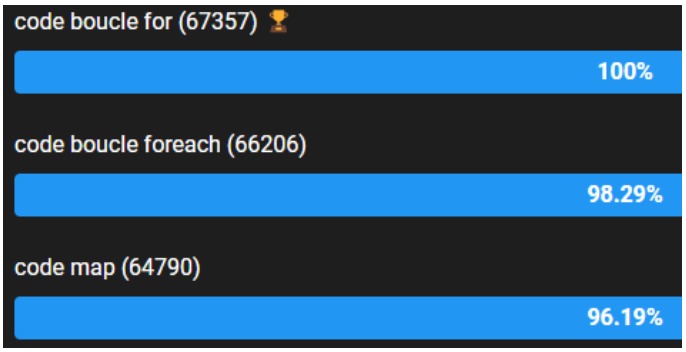
Résultat des tests sur une base de 50 recettes avec tags:

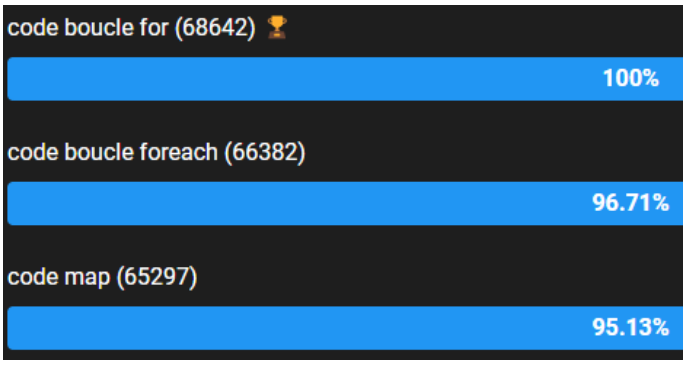
Sur une dizaine de run, l’option 1 avec les boucles natives a été plus rapide à chaque fois (100%)





Résultat des tests for vs map vs forEach:
Sur une dizaine de run, l’option 1 avec les boucles natives a été plus rapide à 8 fois (80%)

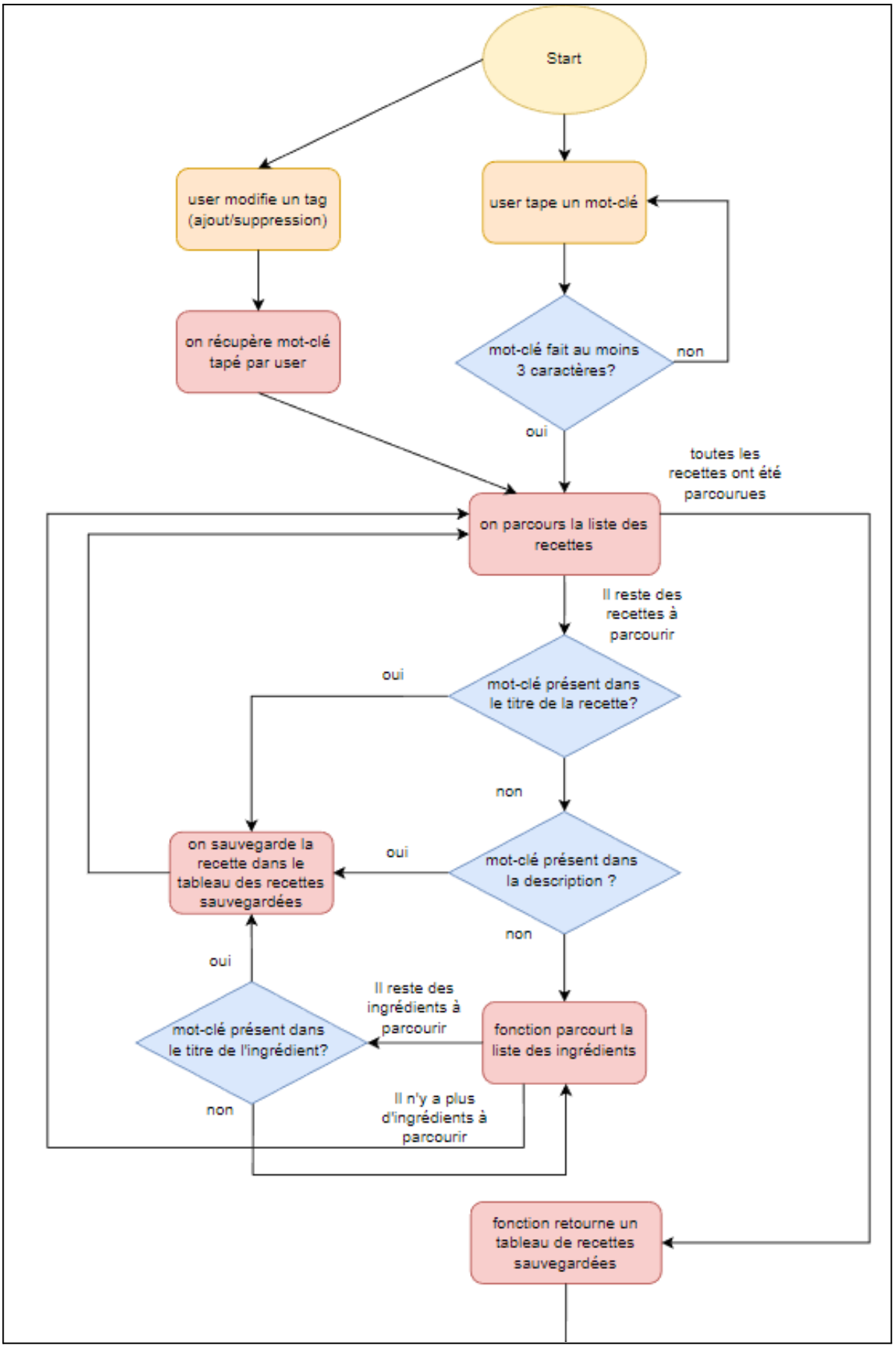




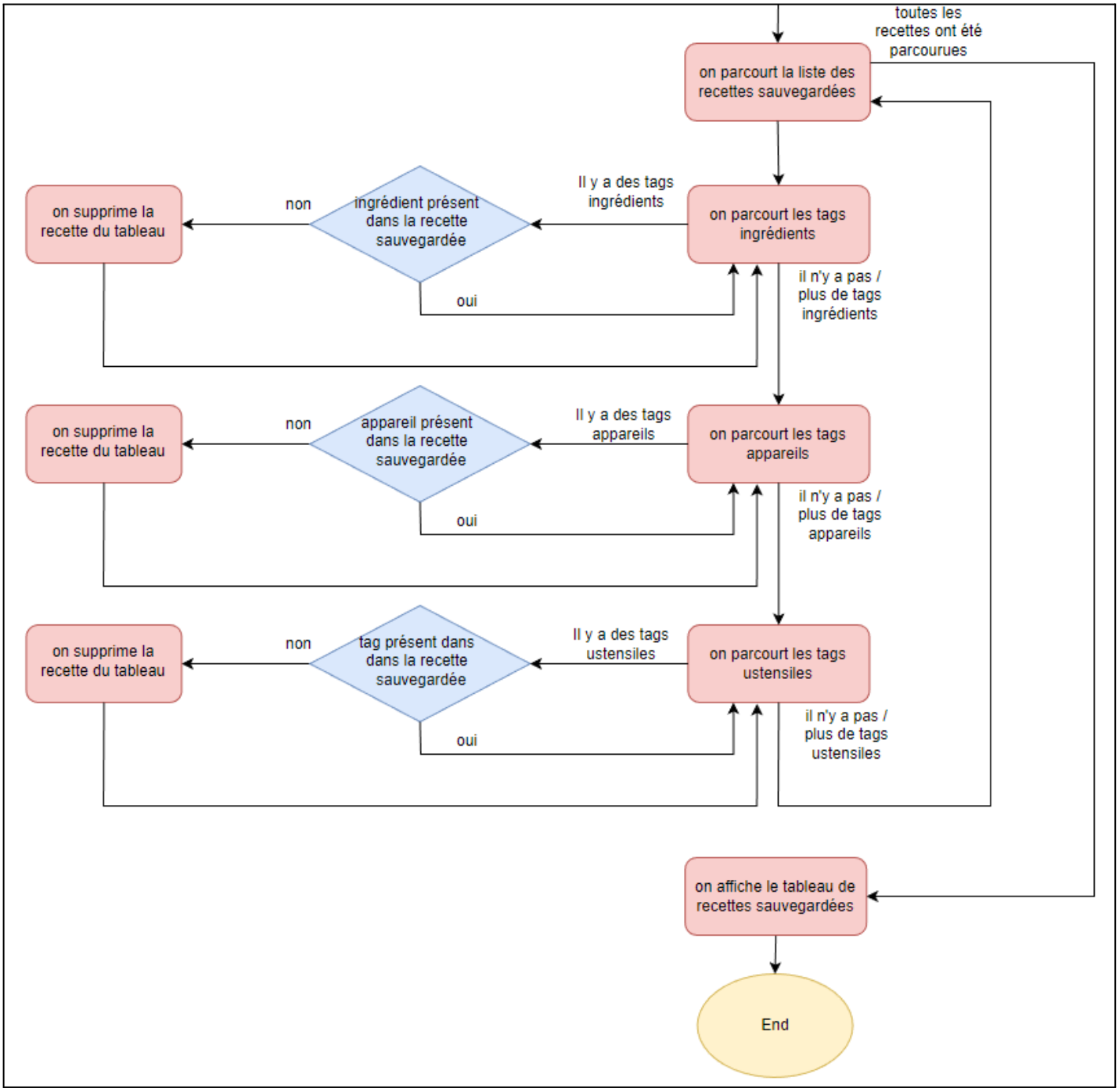
solution retenue:

Au résultat des tests et au vu de la problématique centrée sur la rapidité d'exécution de la fonction, l’option 1 sur les méthodes natives est conseillée car plus rapide sur les tests effectuées et plus modulables sur les conditions d'itérations.

Pour rappel, les tests ont été effectués sur une base de 50 recettes. L’écart de performance pouvant être plus important si le nombre de recettes dans la base de données augmente.



Part 1 Algorigramme: recherche par mot-clé



Part 2 Algorigramme: recherche par tags