

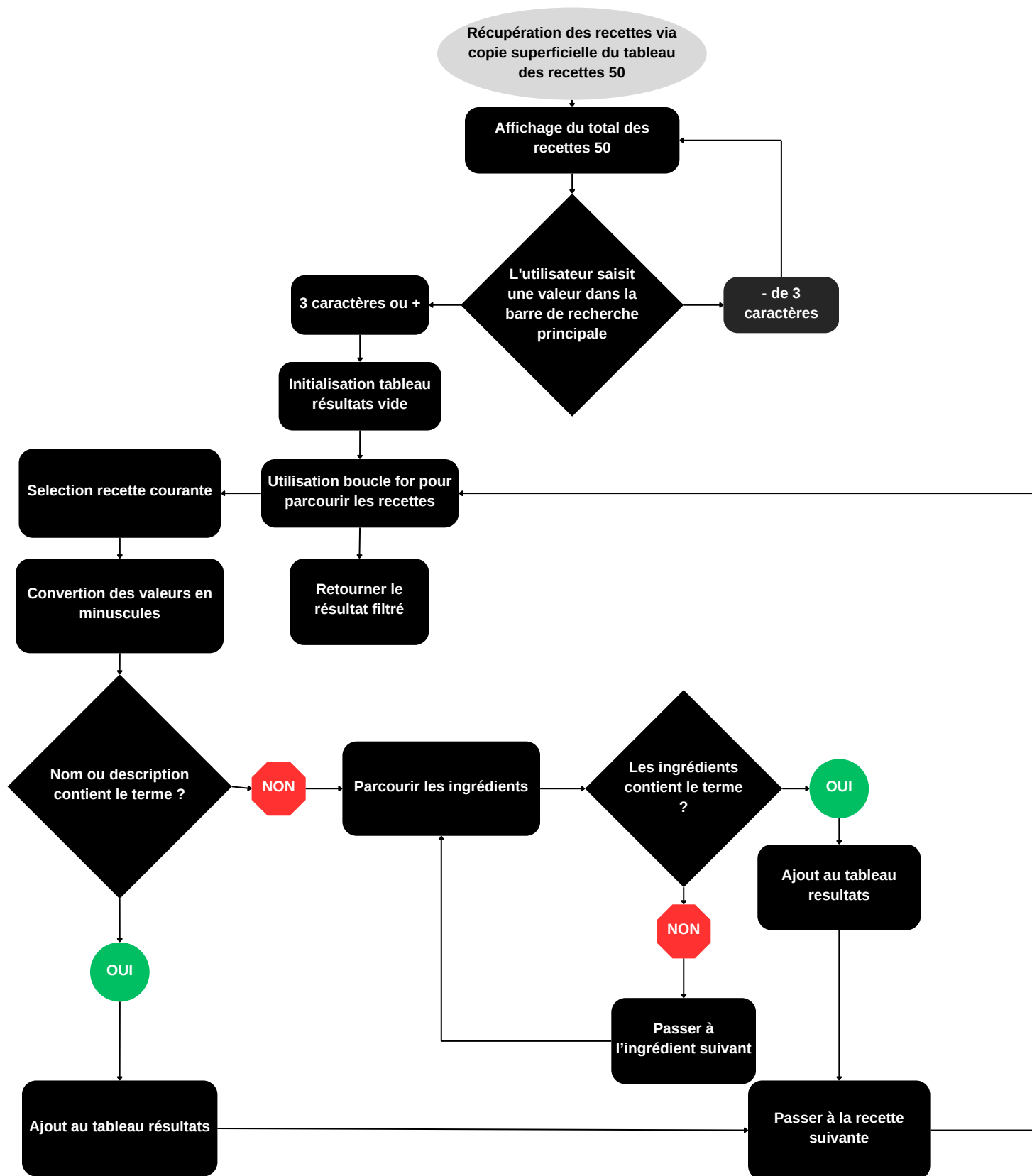
Fiche d'investigation de fonctionnalité

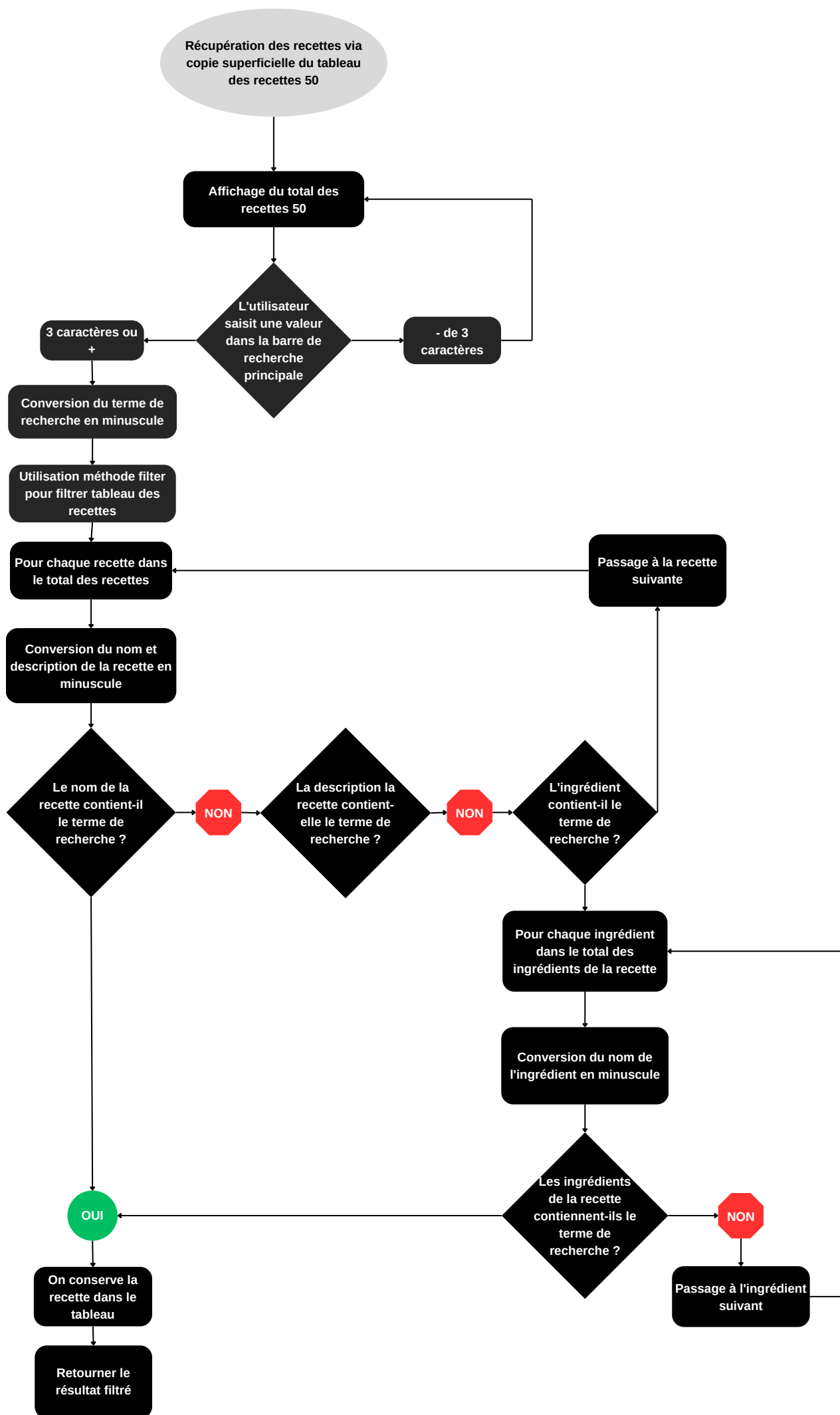
Fonctionnalité : Système de recherche	
Problématique : Pour optimiser les performances de recherche dans l'application Les Petits Plats, nous cherchons à déterminer l'approche la plus efficace entre une implémentation utilisant des méthodes d'array modernes et une implémentation utilisant des boucles natives.	

Option 1 : Méthodes d'Array Modernes Dans cette option, nous utilisons les méthodes d'array modernes comme filter, map, some pour implémenter la recherche.	
Avantages ⊕ Code plus lisible et maintenable ⊕ Syntaxe moderne et concise	Inconvénients ⊖ Consommation mémoire plus importante ⊖ Création de tableaux intermédiaires
Nombre d'itérations sur les données : Multiple (une par méthode chaînée) Création de tableaux temporaires : Oui (à chaque opération) Complexité du code : Faible	

Option 2 : Boucles Natives Dans cette option, nous utilisons des boucles for traditionnelles pour implémenter la recherche. rmulaire (seulement si c'est sa préférence)	
Avantages ⊕ Performances optimales ⊕ Moins d'allocation mémoire ⊕ Une seule itération sur les données ⊕ Pas de création de fonctions intermédiaires ⊕ Meilleure optimisation par le moteur JavaScript	Inconvénients ⊖ Code plus verbeux ⊖ Moins lisible pour les développeurs ⊖ Plus de risques d'erreurs ⊖ Maintenance plus complexe
Nombre d'itérations sur les données : Une seule Création de tableaux temporaires : Un seul Complexité du code : Moyenne	

Solution retenue : L'approche avec les boucles natives a été retenue car elle démontre une supériorité significative en termes de performances lors des tests sur JSBench. Cette solution minimise les opérations de création de tableaux intermédiaires et évite l'utilisation de callbacks, ce qui se traduit par une meilleure gestion de la mémoire et des temps d'exécution optimisés. Bien que cette <u>approche soit moins lisible que l'utilisation des méthodes d'array modernes</u> , l'impact sur la maintenabilité est compensé par une documentation claire et une structure de code logique. Dans le contexte d'une application de recherche en temps réel où la performance est cruciale pour l'expérience utilisateur, les avantages en termes de performances justifient largement ce choix technique.
--







Test JSBEN.CH


JSBEN.CH


BENCHMARKBROWSE



no title  (put title and/or keywords here, which describes your test)


RUN TESTSGENERATE PAGE URLNEW BENCHMARK



Description 


Setup block (useful for function initialization. it will be run before every test, and is not part of the benchmark.) 


Boilerplate block (code will executed before every block and is part of the benchmark. use it for data initializing.) 

native loop  




array method  





result

native loop (4965) 

100%

array method (4712)

94.9%

Votre texte de paragraphe