Fiche d’investigation de fonctionnalité

Fonctionnalité : Recherche des recettes par mot clé / Recherche principale	**Fonctionnalité #2**
Problématique : Afin de satisfaire les utilisateurs, les résultats de la recherche principale doivent apparaître le plus rapidement possible.	

Option 1 : Développer l’algorithme de recherche avec les boucles natives de JavaScript (cf Figure 1)	
Dans cette option, les recettes du tableau sont parcourues par une boucle **For-in**. Les résultats de correspondance avec la saisie utilisateur sont stockés dans des variables dédiées (une pour le résultat de correspondance avec le nom, une pour la description et une dernière pour les ingrédients) et ce, quel que soit le résultat (**null** ou **non-null**). Pour chaque recette, ces trois variables sont comparées et si au moins l’une d’entre elles est à **non-null**, la recette est stockée dans le tableau contenant les recettes à afficher dans l’interface. L’opération se répète jusqu’à avoir parcourues toutes les recettes.	
Avantages : - Utilisation basique de JavaScript - Logique facile à appréhender	**Inconvénients** : - Code lourd - Utilisation de nombreuses variables **(**constantes**)** - Algorithme lent
3 variables et 7 constantes initialisées **2 boucles For-in utilisées** **2 structures conditionnelles utilisées** **Un score au benchmark de 5328 opérations à la seconde**	

Option 2 : Développer l’algorithme de recherche avec les méthodes de l’objet Array de Javascript (cf Figure 2)	
Dans cette option, la méthode **filter()** est appliquée au tableau contenant les recettes. La fonction de test (**callback**) qui lui est assignée contient une structure conditionnelle permettant d’arrêter la recherche à la première correspondance trouvée avec la recherche utilisateur, et ainsi de passer à la recette suivante.	
Avantages : - Code concis - Algorithme de recherche rapide - Facilité de maintient du code	**Inconvénients** : n/a
2 variables initialisées **1 structure conditionnelle contenue dans la fonction callback** **Un score au benchmark de 6167 opérations à la seconde**	

Solution retenue :
L’option 2 qui utilise les méthodes de l’objet Array a été retenue. L’algorithme est 13.6 % plus rapide que la première option. Le code est concis, logique et facilement maintenable.

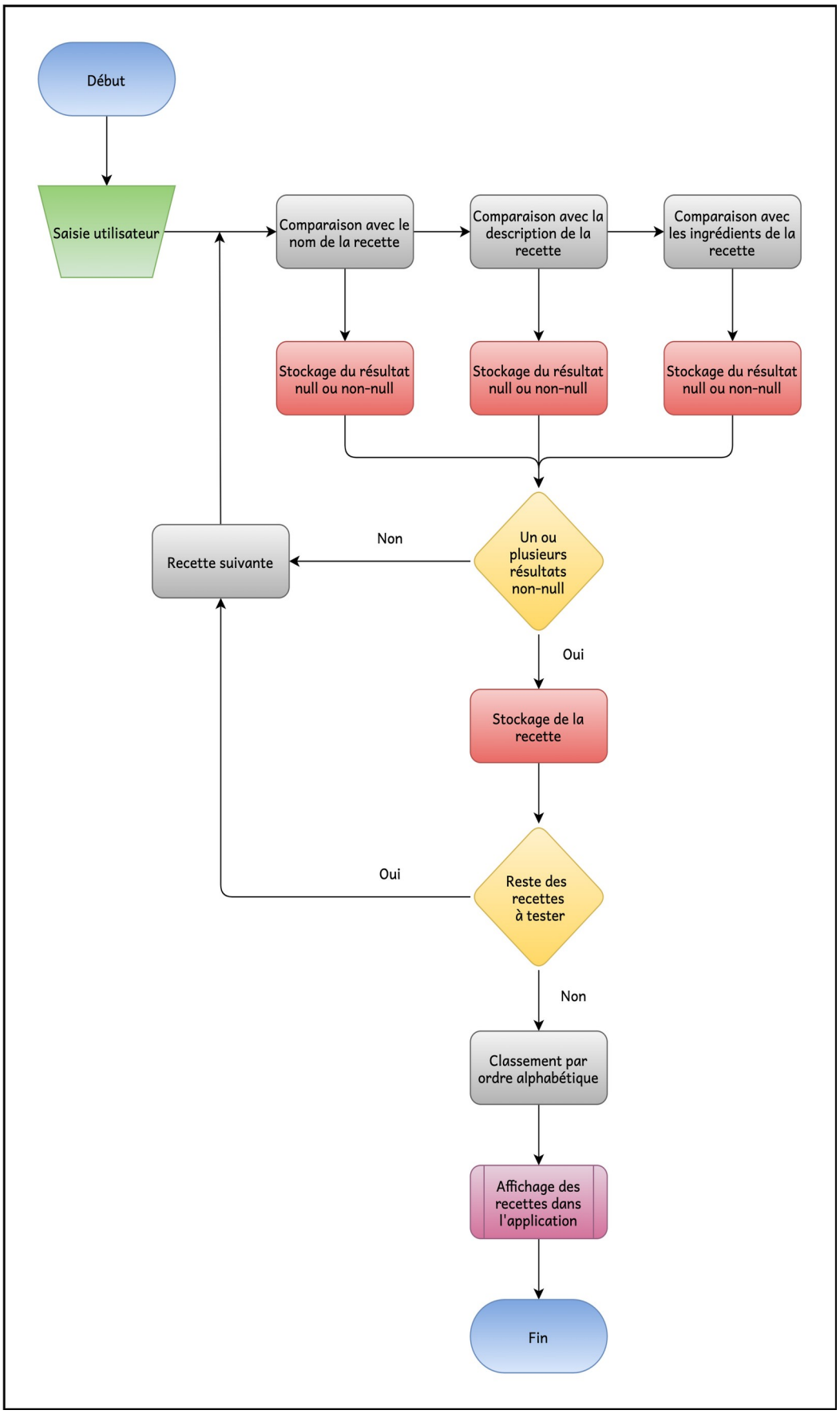

Figure 1 – Algorithme de recherche basé sur les boucles natives JavaScript

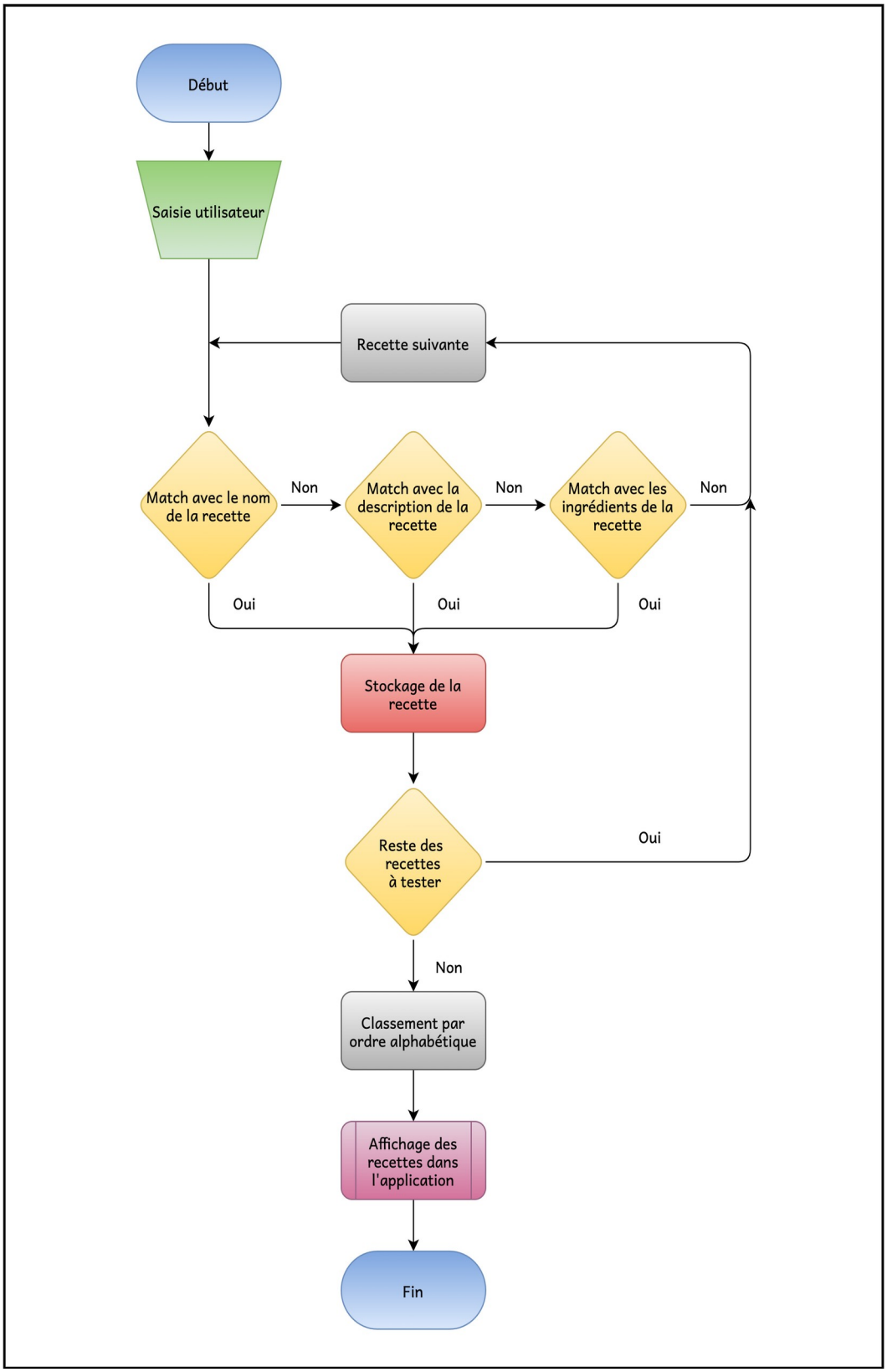

Figure 2 – Algorithme de recherche basé sur les méthodes de l'objet Array JavaScript