

Fiche d'investigation de fonctionnalité

Fonctionnalité : Filtrer les recettes dans l'interface	Fonctionnalité #2
Problématique : Afin de pouvoir retenir un maximum d'utilisateurs, nous cherchons à avoir une séquence de recherche la plus fluide et rapide possible.	

<p>Option 1 : Boucles Natives (cf Figure 1)</p> <p>Dans cette option, on utilise une boucle native qui itère manuellement à travers chaque recette et chaque ingrédient. L'algorithme quitte la boucle d'ingrédients dès qu'un match est trouvé grâce au mot-clé SORTIR DE LA BOUCLE. Cette approche peut être plus lente pour les grands ensembles de données, car elle ne profite pas des optimisations possibles offertes par les fonctions intégrées comme filter.</p>	
<p>Avantages</p> <ul style="list-style-type: none"> ⊕ plus facile à comprendre et à mettre en place 	<p>Inconvénients</p> <ul style="list-style-type: none"> ⊖ lent pour les grands ensembles de données ⊖ ne peut utiliser des optimisations comme filter
Nombre de caractères minimales à entrer pour rechercher: 3	

<p>Option 2 : Programmation Fonctionnelle (cf Figure 2)</p> <p>Dans cette option, on utilise la programmation fonctionnelle. L'algorithme est plus concis et compact, la logique est encapsulée dans des fonctions comme filter et some.</p>	
<p>Avantages</p> <ul style="list-style-type: none"> ⊕ L'utilisation de filter et some permet de profiter des optimisations internes des moteurs d'exécution, ce qui peut conduire à une performance légèrement meilleure. ⊕ Le code est plus court et plus direct, ce qui le rend potentiellement plus facile à maintenir ⊕ Le code est plus court et plus direct, ce qui le rend potentiellement plus facile à maintenir 	<p>Inconvénients</p> <ul style="list-style-type: none"> ⊖ peut être moins intuitif
Nombre de caractères minimales à entrer pour rechercher: 1	

<p>Solution retenue :</p> <p>L'algorithme en programmation fonctionnelle sera plus rapide que celui utilisant des boucles natives car :</p> <ul style="list-style-type: none"> • il possède moins de boucles et d'instructions conditionnelles: L'algorithme avec des boucles natives effectue trois boucles imbriquées : une boucle pour parcourir chaque recette, une boucle pour vérifier chaque ingrédient, et une condition pour vérifier si l'ingrédient correspond. En revanche, l'algorithme en programmation fonctionnelle utilise des fonctions comme filter et some, qui sont optimisées, et permettent d'éviter certaines des étapes explicites des boucles, réduisant ainsi le nombre de passages nécessaires. • son code est plus simple et permet de réduire les erreurs: Boucles natives: Plus de lignes de code et de conditions peuvent introduire des erreurs, notamment lors du traitement des conditions (comme l'utilisation de SORTIR DE LA BOUCLE). <p>Programmation fonctionnelle: Le code est plus concis, les risques d'erreurs sont réduits, et le code est plus facilement optimisable par l'interpréteur ou le compilateur.</p> <p>Conclusion :</p> <p>L'algorithme en programmation fonctionnelle est plus rapide parce qu'il est plus optimisé en termes de gestion des boucles et des conditions, et utilise des fonctions efficaces. De plus, le code est plus simple et compact, ce qui facilite l'optimisation automatique par l'interpréteur ou le compilateur.</p>

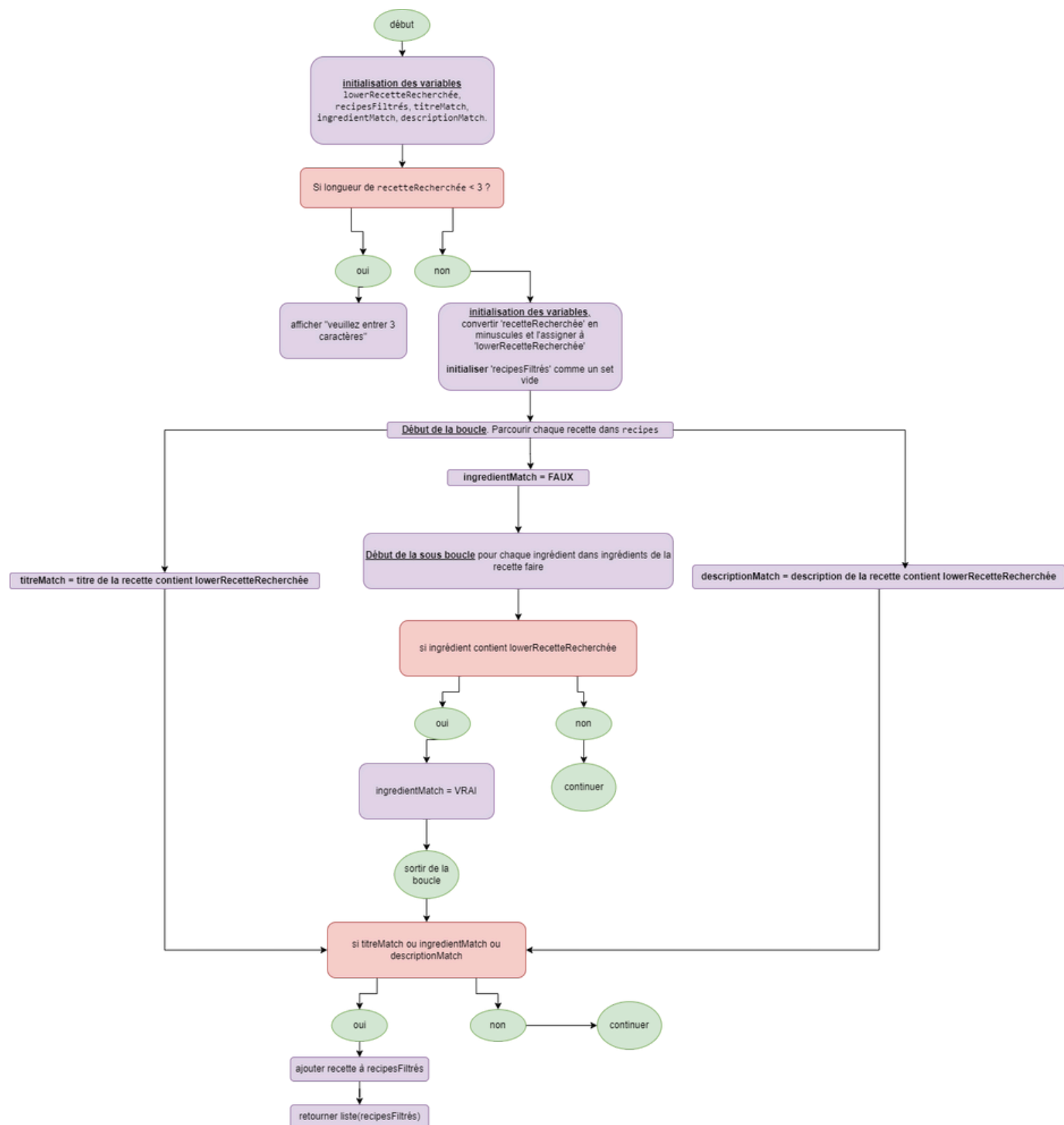


Figure 1: Approche recherche avec boucles natives

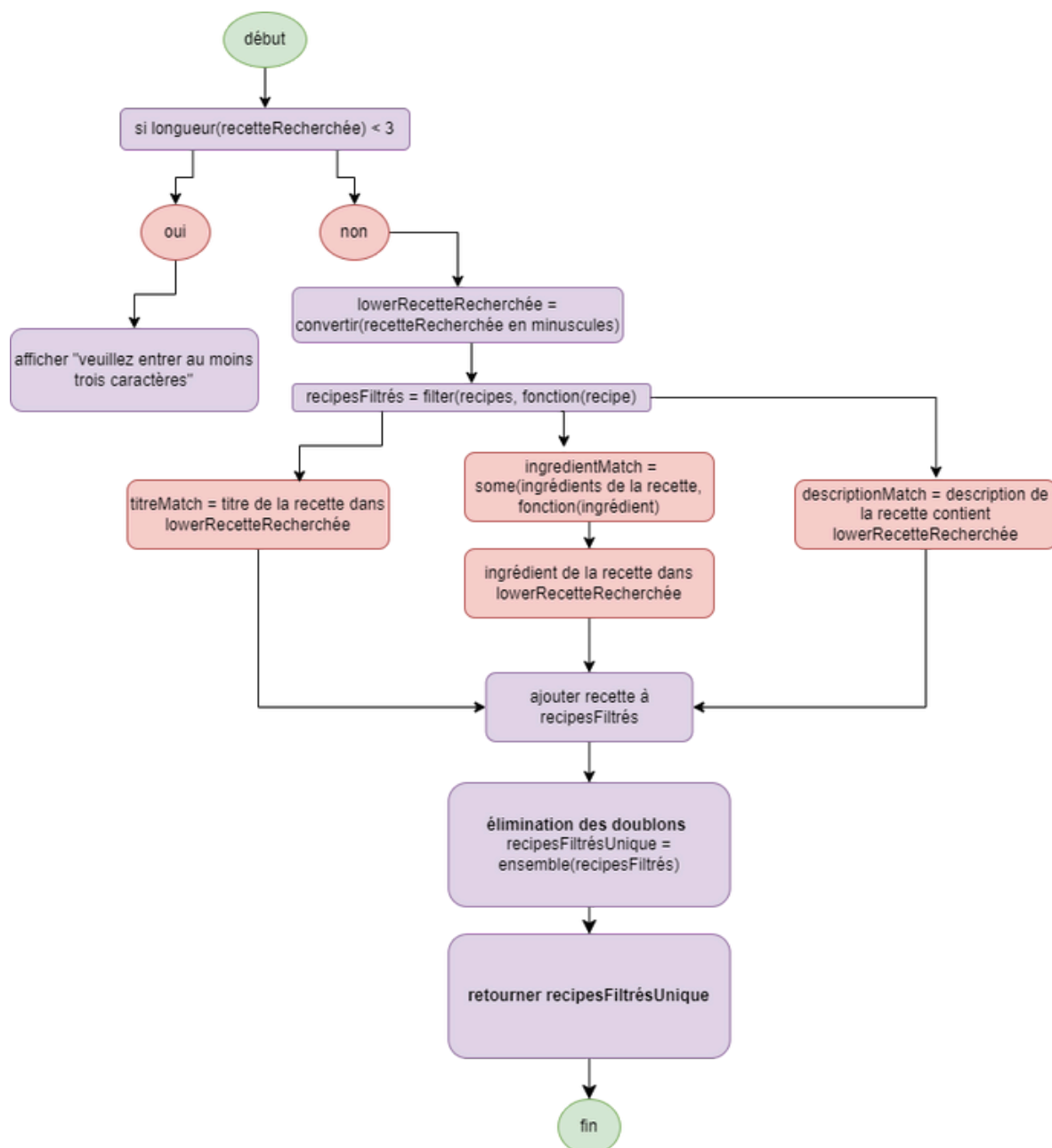


Figure 2 : Approche recherche avec programmation fonctionnelle

résultat de la comparaison des algorithmes obtenu avec jsben.ch

