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.	

Option 1 : Boucles Natives (cf Figure 1)

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.

Avantages	Inconvénients
⊕ plus facile à comprendre et à mettre en place	⊖ lent pour les grands ensemble de données ⊖ moins optimisé

Nombre de caractères minimales à entrer pour rechercher: 3

Option 2 : Programmation Fonctionnelle (cf Figure 2)

Dans cette option, on utilise la programmation fonctionnelle. L'algorithme est plus court et clair, on utilise des fonctions comme filter, some et map.

Avantages	Inconvénients
⊕ L'utilisation des fonctions (filter, some.) permet de meilleures optimisations, ce qui peut donne de meilleures performances. ⊕ Le code est plus court, ce qui le rend potentiellement plus facile à maintenir	⊖ peut être moins intuitif

Nombre de caractères minimales à entrer pour rechercher: 3

Solution retenue :

L'algorithme en programmation fonctionnelle sera plus rapide que celui utilisant des boucles natives car :

- **il possède moins de boucles et d'instructions conditionnelles:** *L'algorithme avec des boucles natives effectue trois boucles : 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).*
  Programmation fonctionnelle: *Le code est plus concis, les risques d'erreurs sont réduits, et le code est plus facilement optimisable.*

Conclusion :
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 son optimisation.

Annexes

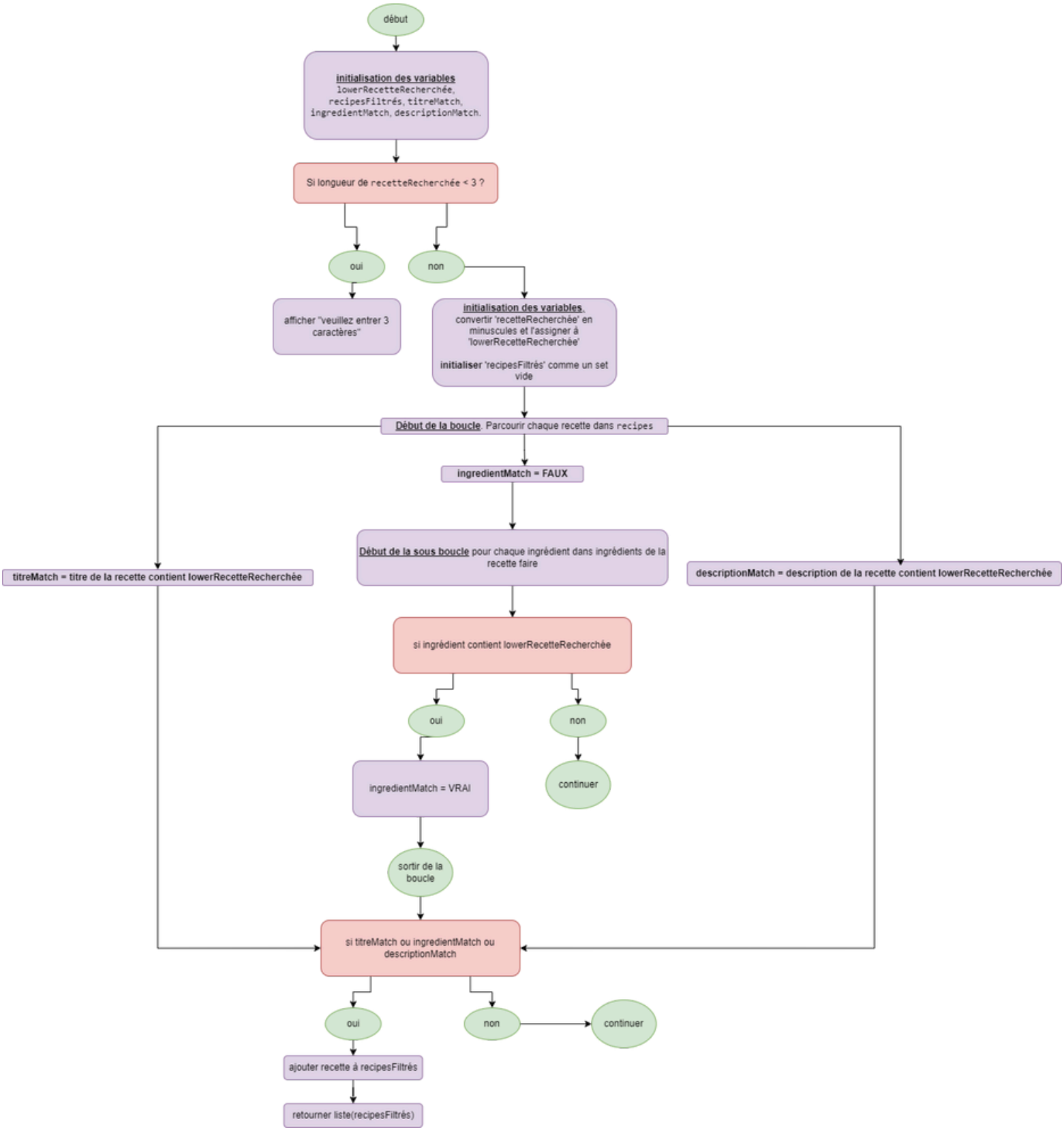

Figure 1: Approche recherche avec boucles natives

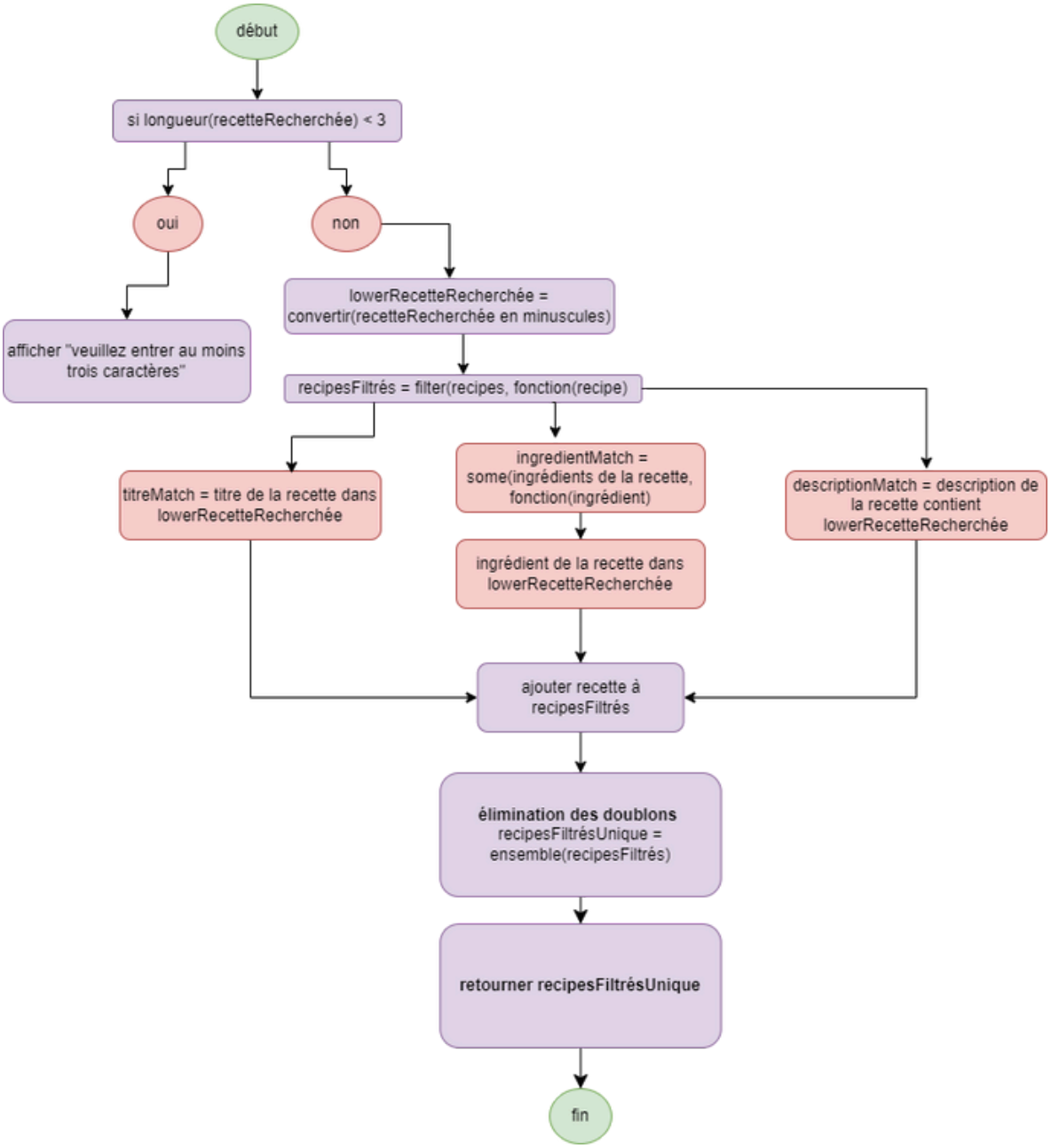

Figure 2 : Approche recherche avec programmation fonctionnelle

résultat de la comparaison des algorithmes obtenu avec jsben.ch