Fiche d’investigation de fonctionnalité

Fonctionnalité : Amélioration de la recherche de recettes	**Fonctionnalité #1**
Problématique : Afin d'optimiser l'expérience utilisateur dans la recherche de recettes, nous explorons des options pour rendre le processus plus efficace en développant une fonctionnalité de recherche permettant aux utilisateurs de trouver rapidement des recettes.	

Option 1 : L'Option 1 privilégie une approche de programmation fonctionnelle en utilisant les méthodes natives de l'objet array telles que foreach, filter, map et reduce.. En utilisant des approches modernes pour le traitement de tableaux, cette option vise à rendre le code plus élégant et expressif. Cependant, il est important de noter que cette approche dépend fortement des méthodes de l'objet array, ce qui pourrait entraîner des problèmes de compatibilité avec certains navigateurs plus anciens.

Avantages	**Inconvénients**
Programmation fonctionnelle avec les méthodes de l'objet array (foreach, filter, map, reduce), offrant un code plus succinct et déclaratif. Utilisation d'approches modernes pour le traitement de tableaux, ce qui peut rendre le code plus lisible.	Dépendance aux méthodes de l'objet array, ce qui pourrait entraîner des problèmes de compatibilité avec certains navigateurs plus anciens. Possibilité de performance légèrement inférieure par rapport à l'approche utilisant des boucles natives.

Option 2 :

L'Option 2 adopte une approche plus classique en utilisant des boucles natives telles que while et for... pour effectuer les opérations nécessaires. Cette méthode offre une approche éprouvée et une compatibilité avec une large gamme de navigateurs, grâce à l'utilisation de fonctionnalités JavaScript standard.

Avantages	**Inconvénients**
Utilisation de boucles natives (while, for...), offrant une approche classique mais éprouvée. Compatibilité avec une large gamme de navigateurs en raison de l'utilisation de fonctionnalités JavaScript standard.	Possibilité de code plus verbeux et moins lisible en raison de l'utilisation de boucles traditionnelles. Moins de fonctions prédéfinies pour le traitement de tableaux, ce qui pourrait augmenter la complexité du code.

Solution retenue : L'Option 1 offre un code plus moderne, déclaratif et lisible, ce qui facilite la maintenance et la compréhension du code. La dépendance aux méthodes de l'objet array peut être gérée en vérifiant la compatibilité des navigateurs ou en utilisant des polyfills au besoin. Cela permet de bénéficier des avantages de la programmation fonctionnelle tout en maintenant une bonne lisibilité du code.

Annexes

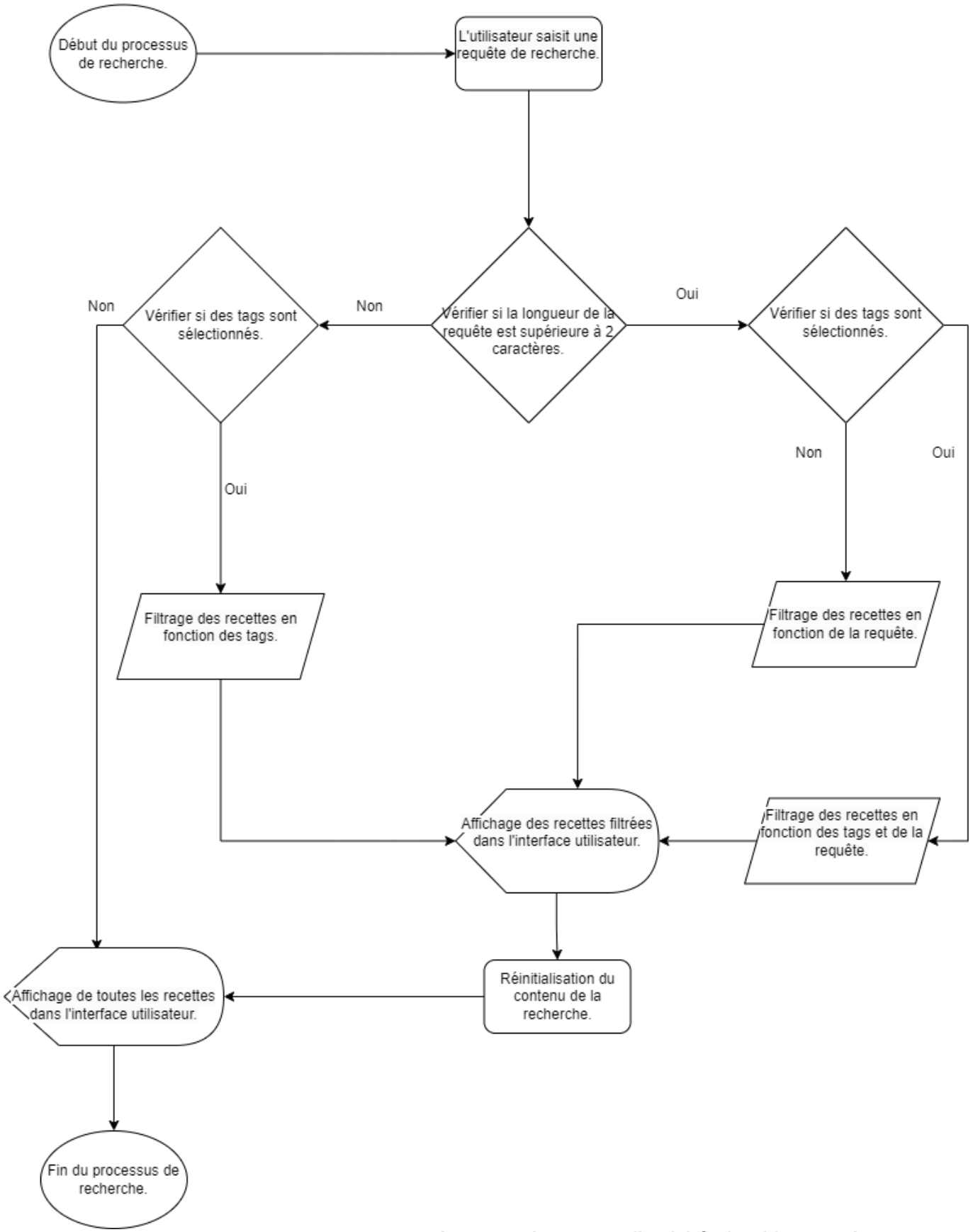

Figure 1 - Diagramme d’activité Algorithme Recherce Recettes

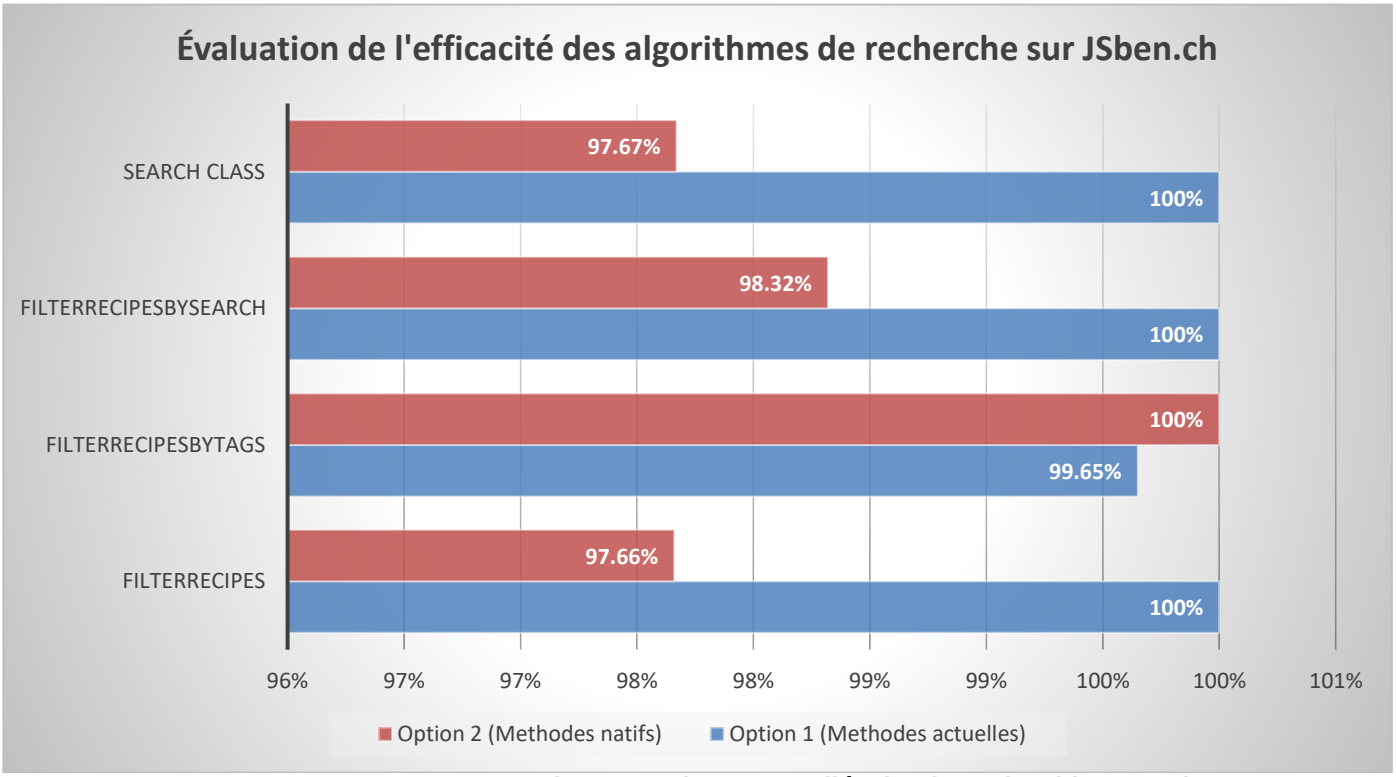

Figure 2 - Diagramme d'évaluation Algorithme Recherce Recettes

Solution Retenue

Suite à l'analyse de l'algorithme de recherche et de ses fonctions principales, nous constatons que **l'Option 1** est plus efficace.

Bien que les arguments généraux suggèrent que les fonctions contenant du code avec des boucles traditionnelles sont plus rapides que celles utilisant les nouvelles méthodes de JavaScript (.map, .filter, .reduce, etc.), nous observons que **l'Option 1** est plus efficace que **l'Option 2**.

Cela peut se produire parce que les moteurs JavaScript modernes (par exemple, V8 dans Chrome, SpiderMonkey dans Firefox) sont hautement optimisés et peuvent effectuer des optimisations sur certaines méthodes de tableau, les rendant très efficaces.

Une autre raison est que certaines méthodes de tableau, comme .map et .filter, ont le potentiel d'une exécution parallèle. Les moteurs JavaScript peuvent optimiser ces opérations pour s'exécuter en parallèle, améliorant ainsi les performances.

Il est important de noter que les performances peuvent varier en fonction du cas d'utilisation spécifique, de la taille des données traitées (par exemple, l'efficacité de la fonction **filterRecipesByTags**) et des optimisations effectuées par les différents moteurs JavaScript.