# Fiche d’investigation

**PRÉSENTÉ PAR**
Noel Emmanuel

# Fiche d’investigation

| Fonctionnailté: Recherche mots | |
|---|---|

**Problématique: Comment concevoir et optimiser un algorithme de recherche par mots-clés pour filtrer les recettes de manière extrêmement rapide, tout en maintenant la pertinence des résultats, en minimisant les ressources matérielles nécessaires, et en garantissant une expérience utilisateur fluide, même avec un volume de données croissant ?**

**Option1 Parcours par Boucle native**

**Avantages :**

1. **Contrôle total : Avec une boucle native, vous avez un contrôle total sur le processus de recherche, ce qui vous permet d'implémenter des logiques de recherche personnalisées.**
2. **Compatibilité : Les boucles natives sont généralement disponibles dans tous les langages de programmation, ce qui les rend largement compatibles.**
3. **Performance prévisible : Dans certains cas, une boucle native peut offrir des performances prévisibles, en particulier pour les petits ensembles de données.**

**Inconvénients :**

1. **Complexité : La mise en œuvre d'une recherche complexe peut nécessiter beaucoup de code, ce qui peut rendre le code moins lisible et plus sujet aux erreurs.**
2. **Moins d'expressivité : Les boucles natives peuvent être moins expressives que les méthodes de tableau, ce qui peut compliquer la compréhension du code.**

1. **Remarque sur la complexité : Les boucles natives peuvent être un choix judicieux pour des logiques de recherche très spécifiques, où un contrôle complet est nécessaire. Cependant, il est important de noter que la complexité augmente avec la complexité de la logique de recherche, ce qui peut rendre le code plus difficile à maintenir.**
2. **Recommandation : Utilisez des boucles natives lorsque vous devez effectuer des opérations de recherche hautement personnalisées et que la performance n'est pas un problème majeur. Assurez-vous de documenter clairement votre code pour faciliter la compréhension et la maintenance.**

# Fiche d’investigation

**Option1 Mehtode array**

**Avantages :**

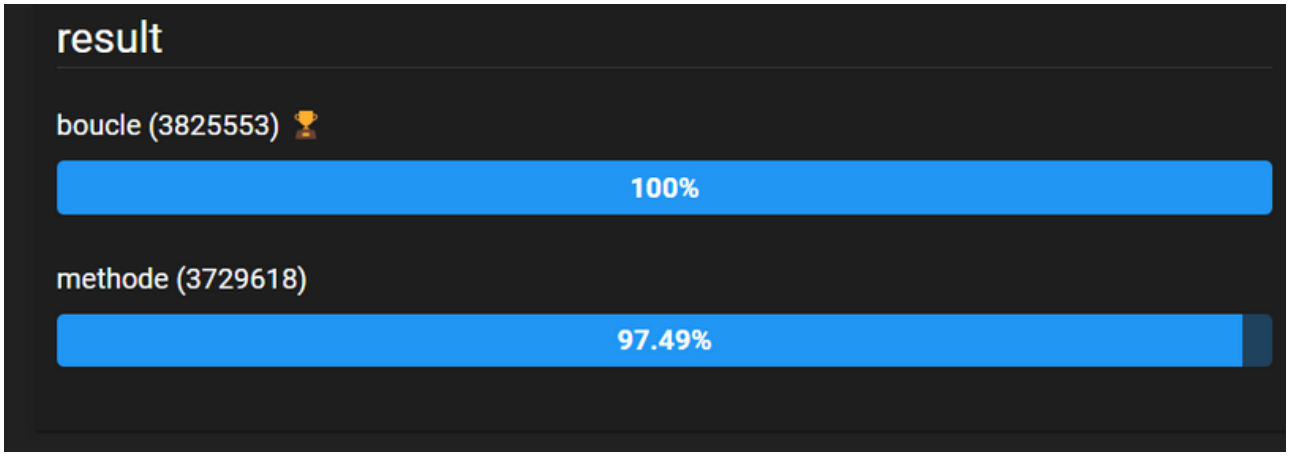
1. **Concision : Les méthodes de tableau permettent d'écrire du code plus concis et lisible pour des opérations de recherche courantes.**
2. **Expressivité : Elles sont souvent plus expressives, ce qui facilite la compréhension du code.**
3. **Optimisation interne : Les méthodes de tableau sont généralement bien optimisées par les bibliothèques standard, ce qui peut améliorer les performances pour de nombreuses opérations de recherche.**

1. **Inconvénients :**
2. **Limitations : Les méthodes de tableau peuvent ne pas être adaptées à des logiques de recherche très complexes. Elles sont principalement conçues pour des opérations de recherche courantes.**
3. **Performance dépendante de l'implémentation : Les performances des méthodes de tableau dépendent de l'implémentation de la bibliothèque standard, ce qui peut varier d'un langage à l'autre.**
4. **Peut nécessiter des opérations intermédiaires : L'utilisation de méthodes de tableau peut impliquer des opérations intermédiaires (comme la création de nouvelles listes) qui peuvent consommer plus de mémoire.**

1. **Remarque sur la concision : Les méthodes de tableau, telles que filter, map, et reduce, permettent d'écrire du code plus concis et lisible pour des opérations de recherche courantes. Elles sont particulièrement utiles pour des cas simples.**
2. **Recommandation : Utilisez des méthodes de tableau lorsque vous devez effectuer des opérations de recherche courantes, et lorsque la lisibilité du code est essentielle. Cependant, soyez conscient que des opérations intermédiaires peuvent être impliquées, ce qui peut avoir un impact sur les performances et la consommation de mémoire. Mesurez les performances si nécessaire.**
3. **Limitations : Les méthodes de tableau peuvent ne pas être la meilleure option pour des logiques de recherche très complexes impliquant des opérations non standard. Dans de tels cas, envisagez des approches alternatives, y compris l'utilisation de boucles natives.**

# Fiche d’investigation

| Solution retenue: |
| --- |
| 1. Après avoir implémenté et testé les deux approches, à savoir les boucles natives et les méthodes de tableau, nous avons déterminé que les boucles natives sont la meilleure option pour notre application de recherche par mots-clés. Voici un résumé des résultats de nos tests :<br>2. Performance : Les boucles natives ont montré des performances supérieures pour les opérations de recherche avec un grand volume de données. L'algorithme de recherche basé sur les boucles natives a été significativement plus rapide que l'approche basée sur les méthodes de tableau dans des scénarios de recherche en temps réel et avec un grand nombre d'opérations par seconde.<br>3. Complexité : Bien que les boucles natives puissent nécessiter plus de code pour implémenter, la complexité n'était pas un problème majeur, et la lisibilité du code a été maintenue grâce à une documentation claire et une structure de code bien organisée.<br>4. Flexibilité : L'utilisation de boucles natives a permis une personnalisation complète de l'algorithme de recherche pour répondre à nos besoins spécifiques sans être limité par les fonctionnalités prédéfinies des méthodes de tableau. |



# Fiche d’investigation