

## Fonctionnalité: Recherche des recettes par tags/texte

**Problématiques:** Effectuer une recherche dans les recettes suivant les données entrées par l'utilisateur. La recherche doit être le plus optimisée possible.

### Option 1: La méthode filter.

Avec la méthode filter sur un tableau de recettes, nous créons un nouveau tableau de recettes avec les recettes dont la vérification de la présence d'un texte, créer sur la base de l'entrée que l'utilisateur à fait dans la barre de recherche textuelle, dans le nom, les ingrédients ou la description renvoie true.

Si on ne trouve nulle part le texte entré par l'utilisateur, on renvoie false et la recette n'est donc pas pris en compte pour la création du nouveau tableau filtré.

### **Avantages:**

- Méthodes 50% plus rapide selon jsben.ch.
- Moins de lignes de codes donc plus facile à lire.
- Le flag permet de facilement passer une recette à false si aucune correspondance n'est trouvée.

### **Option 2: la méthode des boucles natives.**

Nous commençons par créer un tableau vide, nous effectuons ensuite une itération sur les recettes:

Si le texte entré par l'utilisateur est trouvé dans le nom d'une des recettes itérées, on push cette recette dans le tableau. On effectue une nouvelle itération dans les ingrédients et si on trouve le texte de l'utilisateur dans un des ingrédient du tableau ingrédient on push la recette dans le tableau.

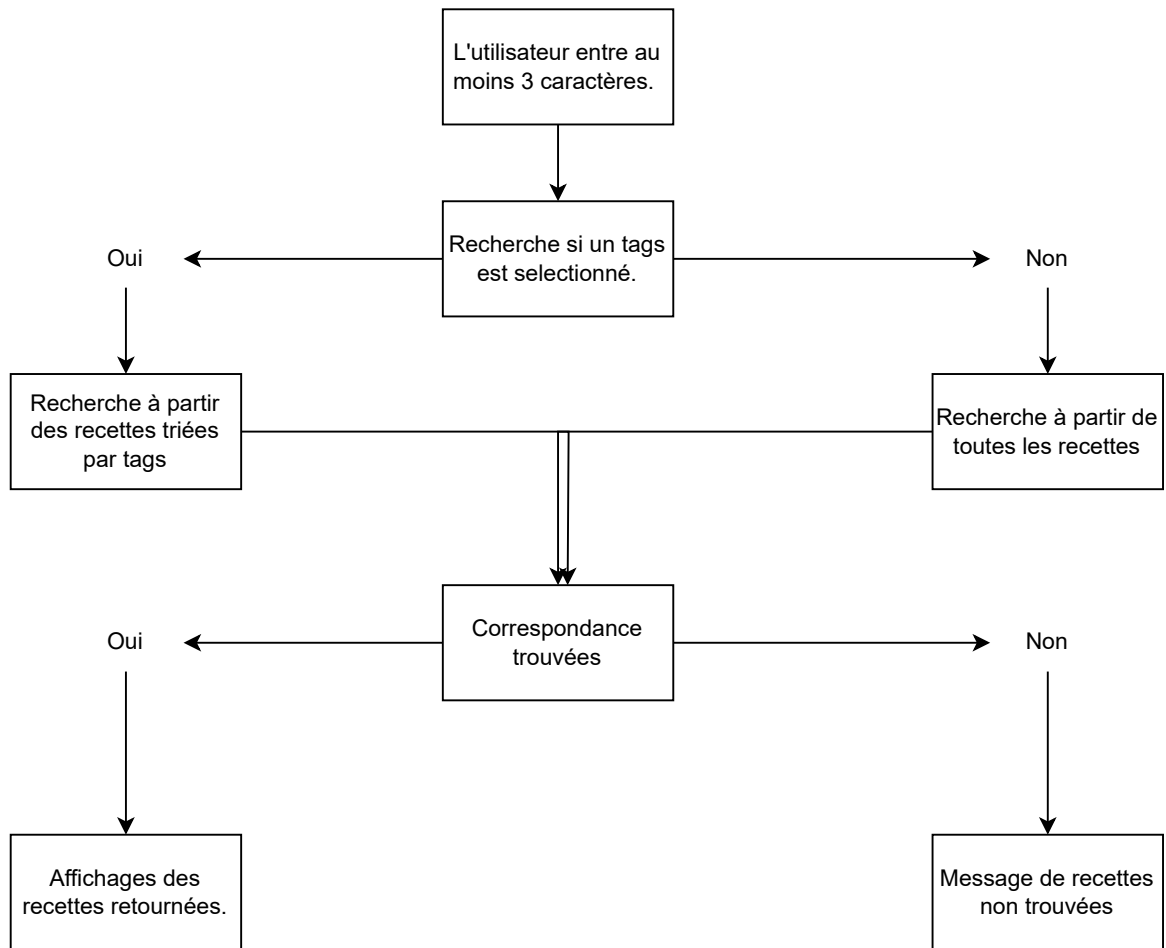
Enfin, si le texte de l'utilisateur est trouvé dans la description on push la recette dans le tableau.

Finalement on effectue une méthode new Set pour supprimer les doublons dans le nouveau tableau.

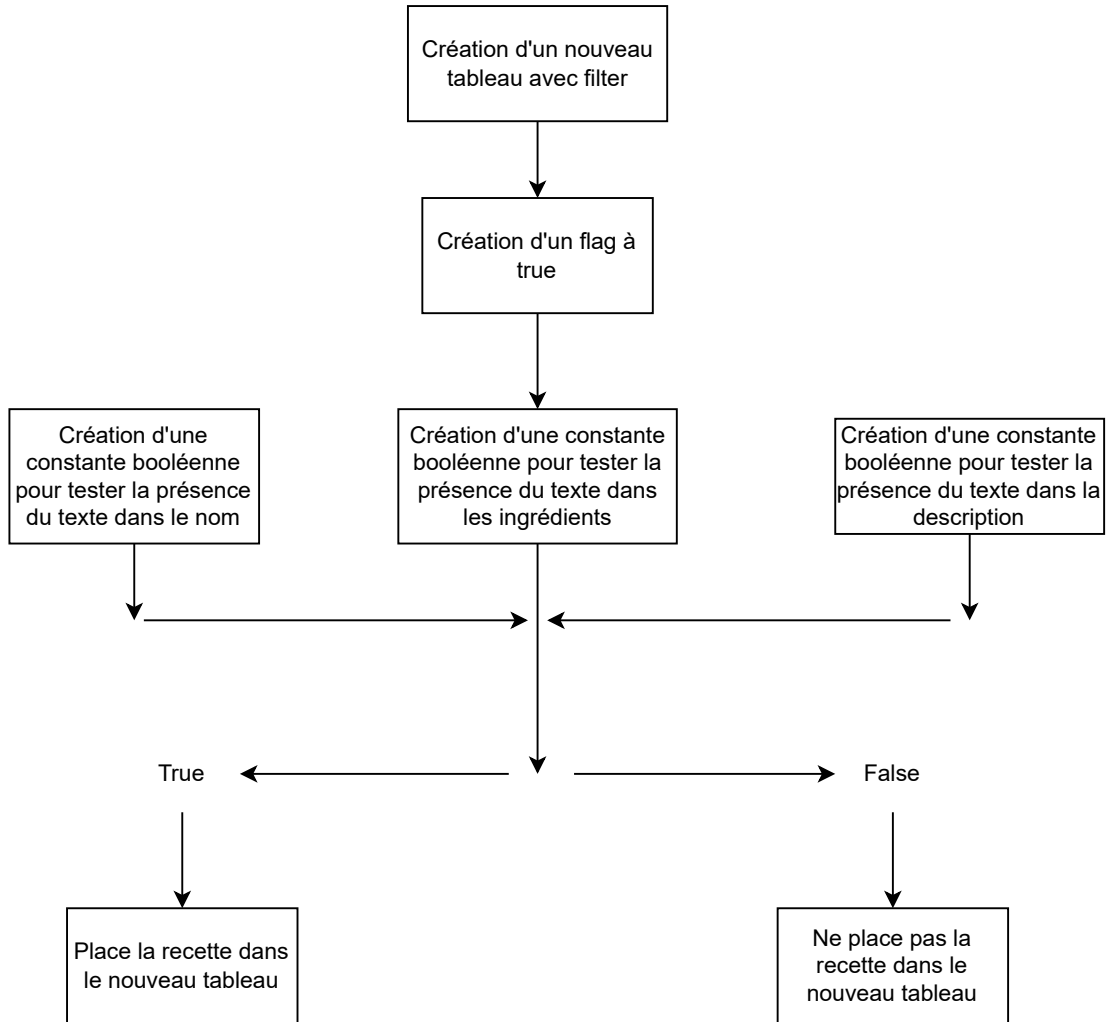
### **Inconvénients:**

- On doit effectuer une opération supplémentaire avec new set pour ne pas avoir de doublons.
- Faire une itération dans une itération pour les ingrédients rend le code moins clair.

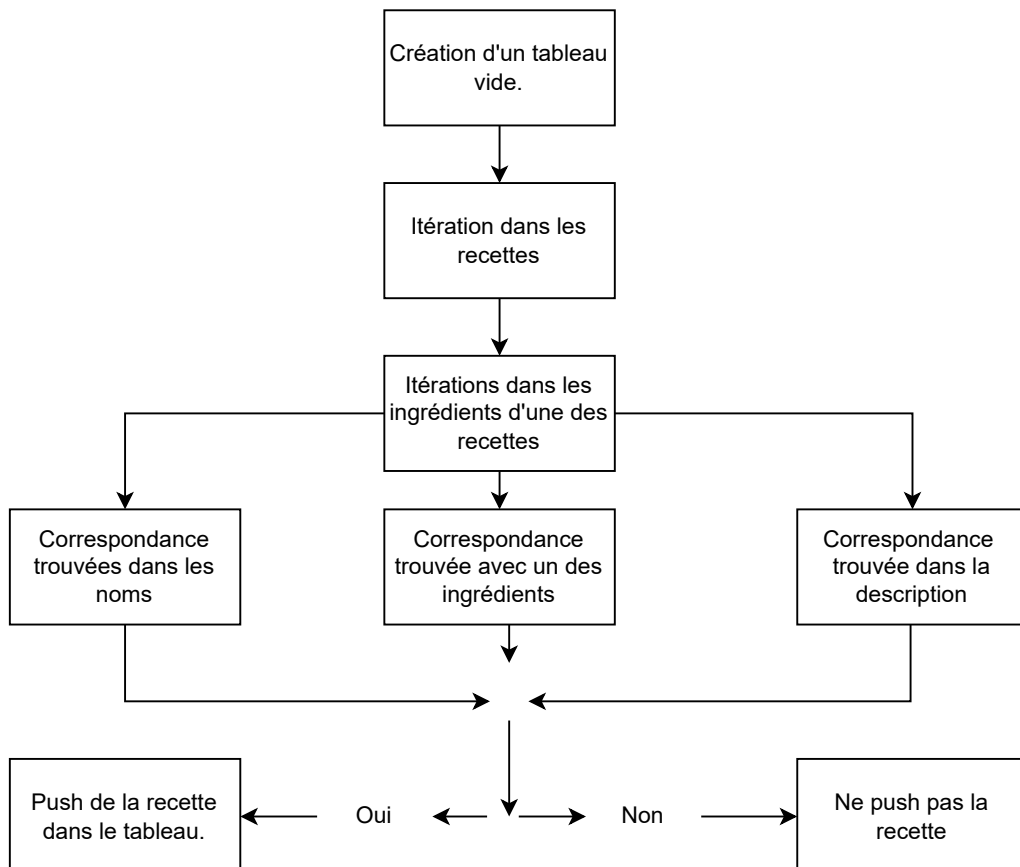
### Méthode de recherche



### Méthode filter



### Méthode boucle native



Setup block (useful for function initialization. It will be run before every test, and is not part of the benchmark.)

boilerplate block (code will executed before every block and is part of the benchmark. use it for data initializing.)

Méthode objets filter

```
1+ recipes.filter((recipe) => {
2+   let isRecipeHasText = true;
3+
4+   ⚠ const isRecipeHasTextInName = textSearch.test(recipe.name.toLowerCase())
5+
6+
7+   ⚠ const isRecipeHasTextInIngredients = recipe.ingredients.some((ingredient) => textSearch.test(ingredient.name.toLowerCase()))
8+   //Si on ne trouve pas d'ingredients on passe le flag à false
9+
10+  ⚠ const isRecipeHasTextInDescription = textSearch.test(recipe.description.toLowerCase())
11+
12+  if (!isRecipeHasTextInName || !isRecipeHasTextInIngredients || !isRecipeHasTextInDescription) {
```

Boucles natives for of

```
1 ⚠ let recipeByTextNative = []
2+   for (let i of recipes) {
3+     if (textSearch.test(i.name.toLowerCase())) {
4+     ⚠ recipeByTextNative.push(i)
5+     }
6+     for (let ingredient of i.ingredients) {
7+       if (textSearch.test(ingredient.name.toLowerCase())) {
```

result







Méthode objets filter (63472) 🏆

100%

Boucles natives for of (41906)

66.02%

If you like to donate (Thank you!):

-  Ethereum (ETH)
-  Chia (XCH)
-  Cardano (ADA)
-  Ravencoin (RVN)
-  Bitcoin (BTC)
-  Ripple (XRP)
-  Litecoin (LTC)
-  Monero (XMR)
-  Dogecoin (DOGE)
-  SOLANA (SOL)