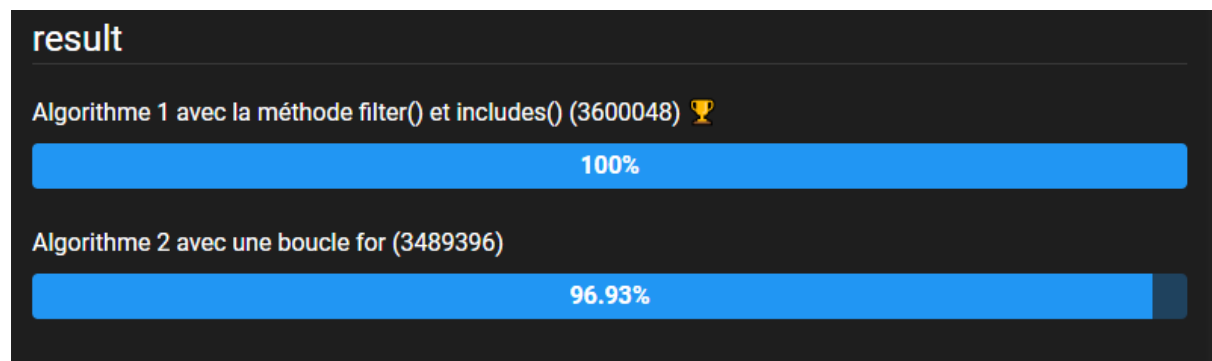


Fiche d'investigation de fonctionnalité

Fiche d'investigation de fonctionnalité	
Fonctionnalité : Algorithme de recherche de recettes	
Problématique : Afin de faire la différence avec les sites de cuisine concurrents nous cherchons à avoir une fonctionnalité de recherche de recettes la plus rapide possible	
Option 1 : Méthode filter() et include()	
Dans cette option, nous utilisons la fonction les méthodes filter et includes pour trier les recettes en fonction de la recherche.	
Avantages	Inconvénients
<i>Code court et facile à comprendre</i>	<i>RAS</i>
<i>Algorithme le plus rapide</i>	
Option 2 : Boucle for()	
Dans cette option, nous utilisons une boucle for pour itérer et trier les recettes en fonction de la recherche.	
Avantages	Inconvénients
<i>RAS</i>	<i>Code plus long et donc + complexe</i>

	Algorithme plus lent
Solution retenue :	
<p>Étant donné que nous recherchons l'algorithme le plus performant et le plus rapide il est préférable de choisir l'algorithme 1 avec la méthode filter et includes car celle-ci est plus rapide selon les tests réalisés sur jsben.ch et jsben.me</p>	

Résultat jsben.ch :



Résultat jsben.me :

<p>algorithme 1 : (méthode filter() et includes())</p> <p>finished</p> <p>916 M ops/s ± 3.68%</p> <p>Fastest</p>	<pre>function search(search) { return this.filtered.filter((recipe) => { return (recipe.name.toLowerCase().includes(search) recipe.description.toLowerCase().includes(search) recipe.ingredients.find((ingredient) => ingredient.ingredient.toLowerCase().includes(search))); }); }</pre>
<p>algorithme 2 : (boucle for)</p> <p>finished</p> <p>846 M ops/s ± 3.11%</p> <p>7.63 % slower</p>	<pre>function searchAlt(search) { let final = []; for (let i = 0; i < this.filtered.length; i++) { let recipe = this.filtered[i]; if (this.match(recipe, search)) { final.push(recipe); } } return final; }</pre>

Schéma de l'algorithme numéro 1 avec la méthode Filter() :

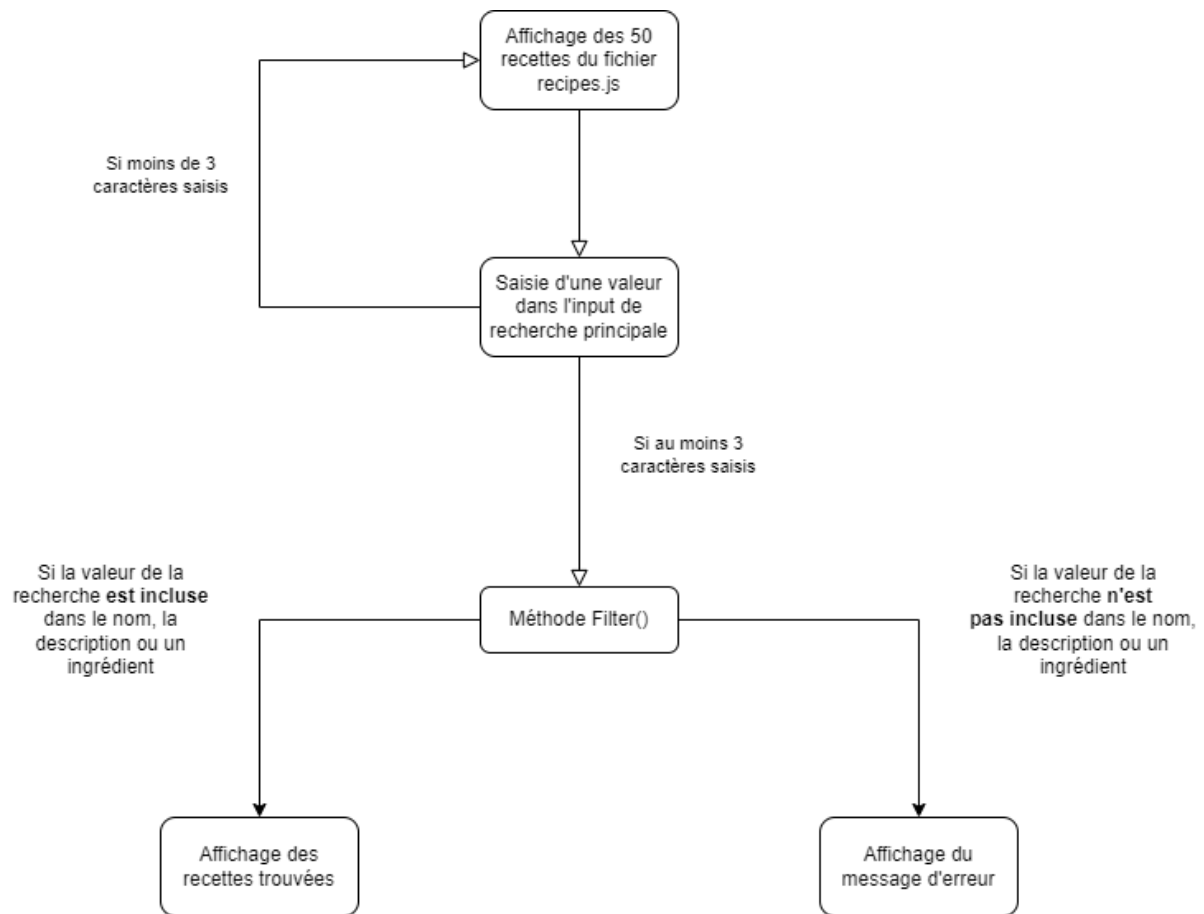


Schéma de l'algorithme numéro 2 avec une boucle For :

