

Fiche d'investigation de fonctionnalité

Fonctionnalité : Moteur de recherche :

Problématique : Les sites offrant des recettes de cuisine sont nombreux et l'équipe a pensé que l'un des éléments qui peuvent faire la différence sur notre site est la fluidité du moteur de recherche.

Option 1 : Algorithme basé sur des boucles natives :

Cette option utilise les boucles « for et while » pour parcourir toutes les données. Dans cette option, nous utilisons une boucle « **do while** » pour exécuter la recherche pour chaque mot de la saisie utilisateur. Puis pour les ingrédients, nous utilisons une boucle « **for** » afin de rechercher, pour chaque ingrédient, s'il contient le mot en question (avec la fonction « **includes** »).

Avantages :	Inconvénients :
Compatibilité avec les anciens navigateurs Optimisé car la boucle est stopée si un des ingrédients contient le mot recherché.	Plus de variables et plus de lignes de codes. Risques d'erreurs accidentelles dues à la complexité du code, Implémentation de nouveau filtre plus lente.

Déroulement du recherche :

L'algorithme est programmé en tant que méthode de l'objet Recette, la méthode est appelée lors d'une boucle « **do while** » sur le tableau des objets Recette. Le déclenchement se fait à partir du 3ème caractère saisie par l'utilisateur dans la barre de recherche principale.

Option 2 : Algorithme basé sur la programmation fonctionnelle

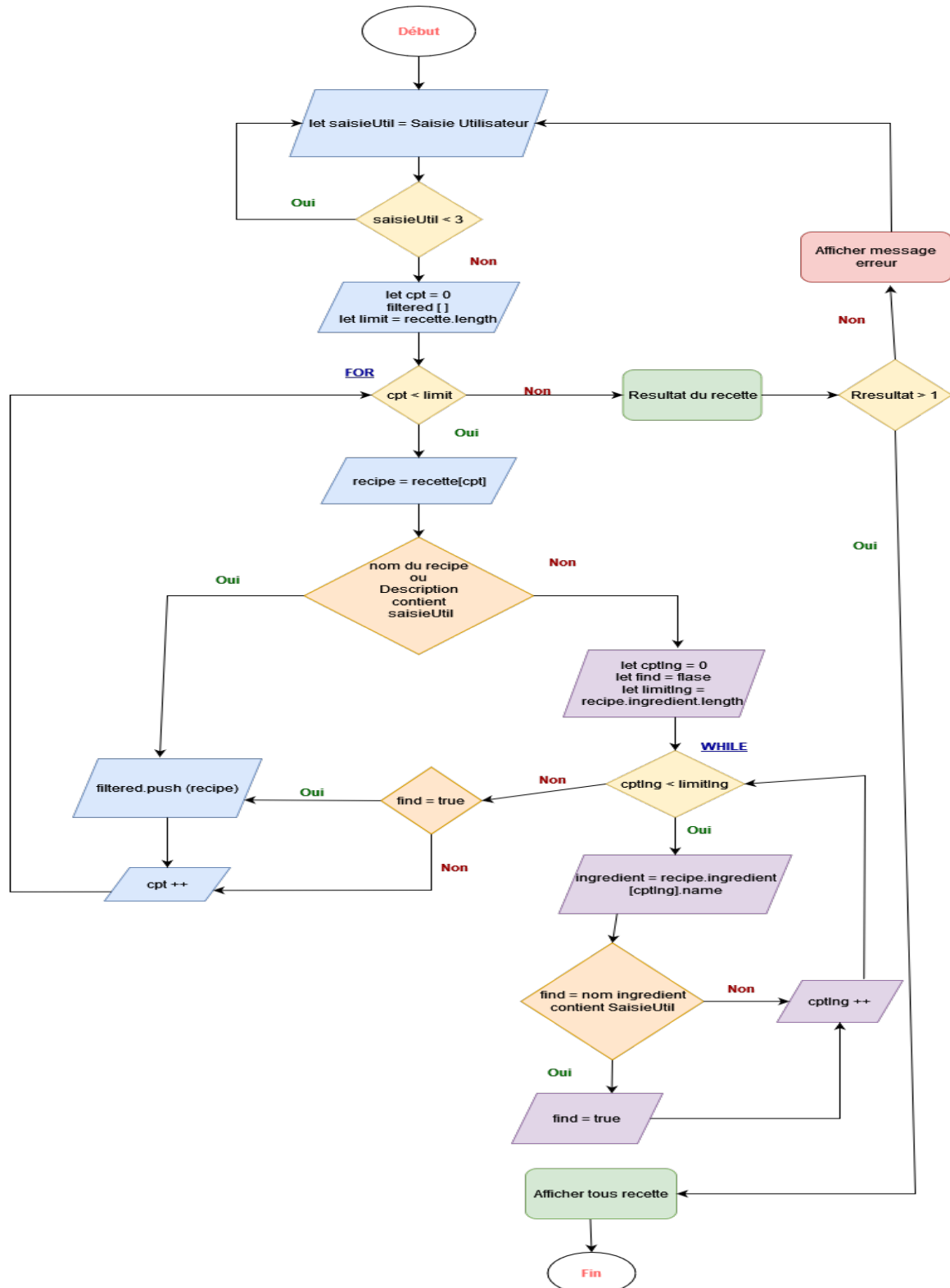
Cette option utilise les standards contemporains de Javascript basé sur les méthodes de l'objet Array « **foreach, filter, map et reduce** ». Dans cette option, nous utilisons la fonction générique d'itérateur « **forEach** » pour exécuter la recherche pour chaque mot de la saisie utilisateur. Puis pour les ingrédients, qui est un tableau d'objet, nous utilisons la méthode « **find** » afin de rechercher, pour chaque ingrédient, s'il contient le mot en question (avec la fonction « **includes** »).

Avantages :	Inconvénients :
Lisibilité du code, Moins de risques d'erreurs accidentelles, Implémentations de nouveau filtre rapide	Compatibilité avec les anciens navigateurs.

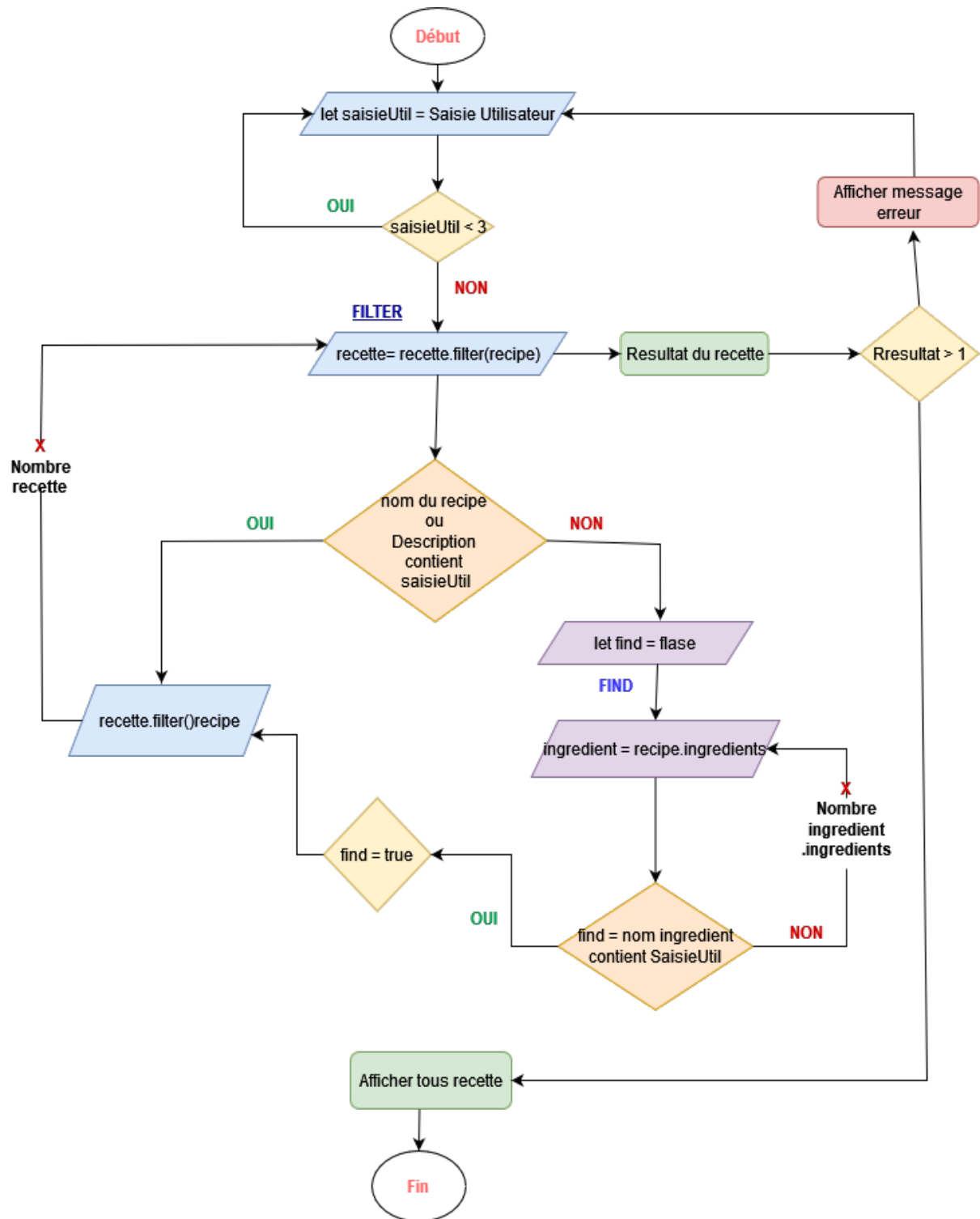
Déroulement de la recherche :

L'algorithme est programmé en tant que méthode de l'objet Recette, la méthode est appelée lors « **forEach** » sur le tableau contenant l'ensemble des objets Recette. Le déclenchement se fait à partir du 3ème caractère saisi par l'utilisateur dans la barre de recherche principale.

Algorithme Option 1 : Native




Algorithme Option 1 : Fonctionnelle



Résultats des tests en utilisant le mot de recherche 'coco'

Solution aux tests réalisés entre les 2 méthodes sur jsben.ch, l'option 2 a été retenue. Elle s'est démarquée par sa rapidité (entre 16 à 18% plus rapide que l'option 1. Et il est clair qu'elle se démarque également par sa maintenabilité et la facilité d'implémenter de nouvelle option de recherche par le futur

 Run Please login/register to save & publish tests Sponsor Settings Sign in

Teste Algorithme Petit Platsv1- by

enter test suite description

Setup HTML - click to add setup HTML

Setup JS - click to add setup JavaScript

<div>Fonctionnelle</div> <div>finished</div> <div>5882.84 ops/s ± 0.74%</div> <div>Fastest</div>	<pre>data.forEach((recipe) => recette.push(new Recipe(recipe))); //prendre tous les donne et mettre dans nouvelle tableau const saisiUtil = 'coco'; //variable saisie utilisateur 'coco' recette = recette.filter((recipe) => { //prendre tous les recette si... if (recipe.name.includes(saisiUtil)) return true; //si le saisie est dans le nom if (recipe.description.toLowerCase().includes(saisiUtil)) return true; // si le saisie est dans description else if (!recipe.ingredients.find((ingredient) => ingredient.name.includes(saisiUtil))) return true; //dans le parcours de chacun ingredient, si la saisie est dans le ingredient return false; });</pre>	<div>DEFER</div>
<div>Native</div> <div>finished</div> <div>4874.77 ops/s ± 2.06%</div> <div>17.14 % slower</div>	<pre>let i=0; while (i < recipe.ingredients.length) { const ingredient = recipe.ingredients[i].name; if (cherche.includes(ingredient)) { filtered.push(recipe); break; } i++; }</pre>	<div>DEFER</div>

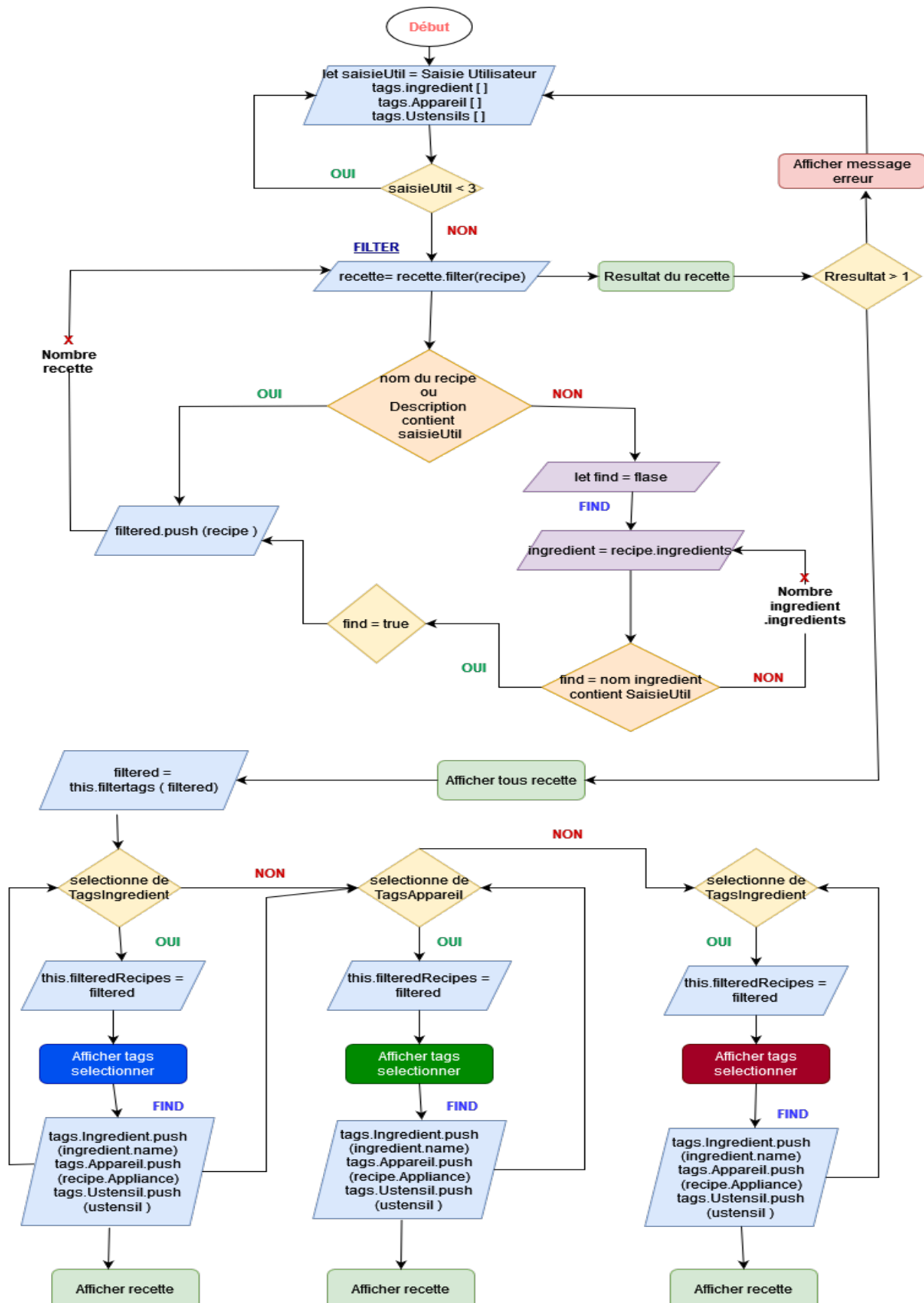
Test Case - click to add another test case

Teardown JS - click to add teardown JavaScript

Output (DOM) - click to monitor output (DOM) while test is running

RUN again

Recherche Globale :



Recherche directement sur les tags

