Fiche d'investigation fonctionnalité #1

Fiche d'investigation de fonctionnalité

Fonctionnalité : Recherche et tri des recettes	**Fonctionnalité #1**
Problématique : Afin d'accéder rapidement à une recette, il faut trouver le meilleur algorithme de tri filtrant les recettes. L'algorithme choisi doit être le plus performant car les utilisateurs veulent une recherche rapide, quasi instantanée et donc plus concurrentielle.	

Option 1 : **Algorithme avec des boucles natives (boucle for)** À partir du mot entré dans l'input, la recherche est effectuée en itérant chaque recette pour touver les correspondances avec le nom ou la description ou les ingrédients.	
Avantages - Les boucles natives permettent de répéter des actions simpement. - Elles sont très modulables et facilement utilisables	**Inconvénients** - Le code est plus verbeux - Maintenabilité - Les boucles natives deviennent plus lentes lorsqu'ells sont imbriquées
Nombre de champs minimum à remplir pour une recherche : 3	

Option 2 : Algorithme avec la programmation fonctionnelle Le tableau des recettes est filtré avec la méthode filter() et avec une fonction callback, des correspondances sont recherchées dans le nom, la description ou les ingrédients.	
Avantages Le code est plus court, plus lisible, plus simple à réaliser	**Inconvénients** Modulabilité
Nombre de champs minimum à remplir pour une recherche : 3	

Solution retenue :
C'est l'option 2 qui a été retenu. Même si la différence entre les deux algorithmes n'est pas immense, nous pensons que si la base de données était plus étendues, ladifférence serait notable.

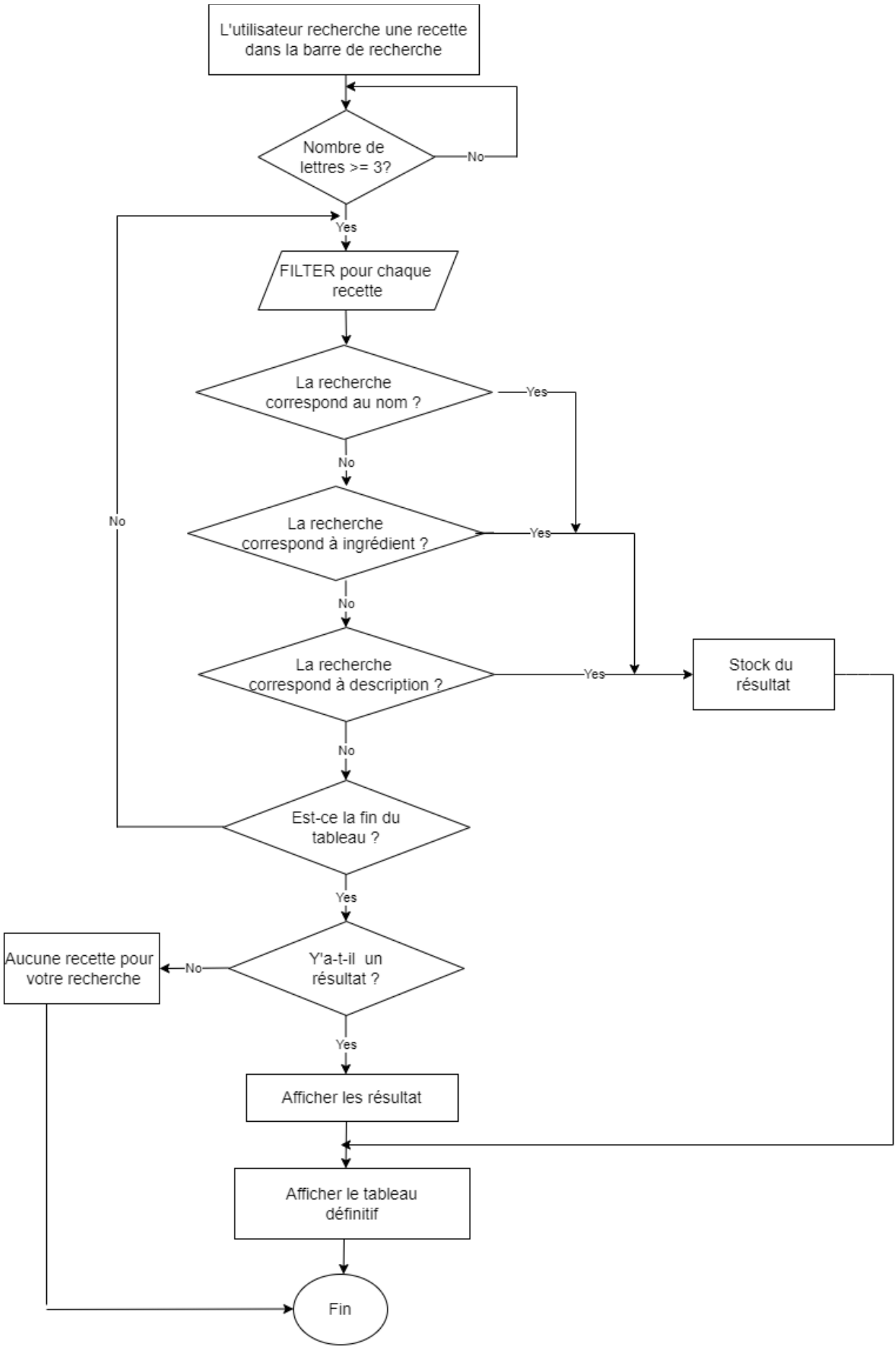

Diagramme 2 : Programmation fonctionnelle avec Filter

JSBench.Me ▶ Run Please login/register to save & publish tests

Setup HTML - click to add setup HTML

Setup JavaScript

```
let a = {
    "recipes": [
        {
        "id": 1,
        "name" : "Limonade de Coco",
        "servings" : 1,
        "ingredients": [
            {
                "ingredient" : "Lait de coco",
                "quantity" : 400,
                "unit" : "ml"
            },
            {
                "ingredient" : "Jus de citron",
                "quantity" : 2
            },
            {
                "ingredient" : "Crème de coco",
                "quantity" : 2,
```

Boucle native For

finished

1018502950.74 ops/s ± 1.2%
4.03 % slower

```
let query = "frangipane"
const filteredRecipes = (recipes, query) => {

    const results = [];

        if (result) {

        for (let i = 0; i < result.length; i++) {

            const { name, description, ingredients } = result[i];
            if (name.toLowerCase().includes(query) || description.toLowerCase().includes(query) ) {
                results.push(result[i]);
                continue;
            }

            for (let i = 0; i < ingredients.length; i++) {
                if (ingredients[i].ingredient.toLowerCase().includes(query)) {
                    results.push(result[i]);
                    break;
```

Programmation fonctionnelle avec Filter

finished

1061320221.45 ops/s ± 1.05%
Fastest

```
let query = "frangipane"
const filteredRecipes = (recipes, query) => {
    if (result) {

        return result.filter((result) => {
            return (
                result.name.toLowerCase().includes(query) ||
                result.description.includes(query) ||
                result.ingredients.some((ingredient) => ingredient.ingredient.includes(query))
            );

        });

    } else {

        return recipes.filter((recipe) => {
            return (
                recipe.name.toLowerCase().includes(query) ||
                recipe.description.includes(query) ||
```