Fiche d'investigation de fonctionnalité

Fonctionnalité : Recherche	**Recherche principale**

Problématique :

Cette algorithme doit fonctionner avec un enchainement de 2 étapes de recherche comprenant la recherche principale et les filtres. Pour tout algorithme important qu'on développe, on a pour habitude d'en faire deux implémentations différentes pour pouvoir comparer leurs performances et choisir la meilleure.

Option 1 : **Input Search V1 - méhode fonctionnelle**

Dans cette option, on utilise une programmation fonctionnelle en manipulant des objets array(filter, map, some …) afin d'optimiser le fonctionnement des algorithmes, ainsi que pour avoir plus de réactivité lors des recherches

Avantages :	**Inconvénients :**
Algorithme plus rapide Fichiers plus léger Maintenance plus facile	Moins lisible

Informations :

Une saisie dans la barre de recherche déclenche une recherche parmi : le nom, les ingrédients et la description de la recette. La sélection d'un ou plusieurs tag(s) parmi trois catégories (ingrédients, ustensiles, appareils), déclenche la recherche dans les clés "ingredient", "ustensils" et "appliances" de chaque objet recette.

Option 2 : **Input Search V2 - méthode impérative**

Dans cette option on utilise des boucles natives (for) pour la recherche principale

Avantages :	**Inconvénients :**
Code plus lisible	Moins rapide

Informations :

Séquence d'instructions composée :

de boucles `for`

de structures conditionnelles `if`

Solution retenue :

Bien que l'échantillon de 50 recettes soit assez réduit pour comparer de manière fiable les différences de performances entre les deux algorithmes, l'approche fonctionnelle (option 1) semble plus efficace que l'approche impérative. Nous retenons donc l'option 1,

Les petits plats - Search V1

finished

606 M ops/s ± 2.85%
Fastest

```
function inputSearch(value, recipes) {
  //sur deuxième branche interdit filter, find, map, foreach, findindex, includes, etc...
  let recipesFiltred = [...recipes];

  if (value.length >= 3) {
    recipesFiltred = recipes.filter(
      (recipe) =>
        recipe.name.toLowerCase().match(new RegExp(value, "i")) ||
        recipe.description.toLowerCase().match(new RegExp(value, "i")) ||
        recipe.ingredients.some((ingredient) =>
          ingredient.ingredient.toLowerCase().match(new RegExp(value, "i"))
        )
```

Les petits plats - Search V2

finished

570 M ops/s ± 9.07%
Fastest

```
function inputSearch(value, recipes) {
  //Algorithme V2
  //deuxième algorithme sans filter, find, map, foreach, findindex, includes, some, etc...

  let recipesFiltred = [];

  if (value.length >= 3) {
    for (let i = 0; i < recipes.length; i++) {
      const recipe = recipes[i];
      const nameMatch = recipe.name.toLowerCase().match(new RegExp(value, "i"));
      const descriptionMatch = recipe.description
        .toLowerCase()
```

Search V1	606 M ops/s	La V1 est plus rapide que la V2
Search V2	570 M ops/s	

JSBEN.CH — BENCHMARK | BROWSE | DONATE

Les petits plats

RUN TESTS | GENERATE PAGE URL | NEW BENCHMARK

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

```
            unit: "grammes",
          },
        ],
        time: 60,
        description:
          "Préparer la frangipane : Mélanger le sucre la poudre d'amande
        appliance: "Four",
        ustensils: ["rouleau à patisserie", "fouet"],
      },
    ];
    const value="mix";
```

ADD LIBRARY

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

Search V1

```
function inputSearch(value, recipes) {
  // Algorithme V1

  let recipesFiltred = [...recipes];

  if (value.length >= 3) {
    recipesFiltred = recipes.filter(
      (recipe) =>
        recipe.name.toLowerCase().match(new RegExp(value, "i")) ||
        recipe.description.toLowerCase().match(new RegExp(value, "i")) ||
        recipe.ingredients.some((ingredient) =>
```

Search V2

```
function inputSearch(value, recipes) {
  //Algorithme V2
  //deuxième algorithme sans filter, find, map, foreach, findindex, include

  let recipesFiltred = [];

  if (value.length >= 3) {
    for (let i = 0; i < recipes.length; i++) {
      const recipe = recipes[i];
      const nameMatch = recipe.name.toLowerCase().match(new RegExp(value,
      const descriptionMatch = recipe.description
        .toLowerCase()
```

result

Search V1 (3014500)

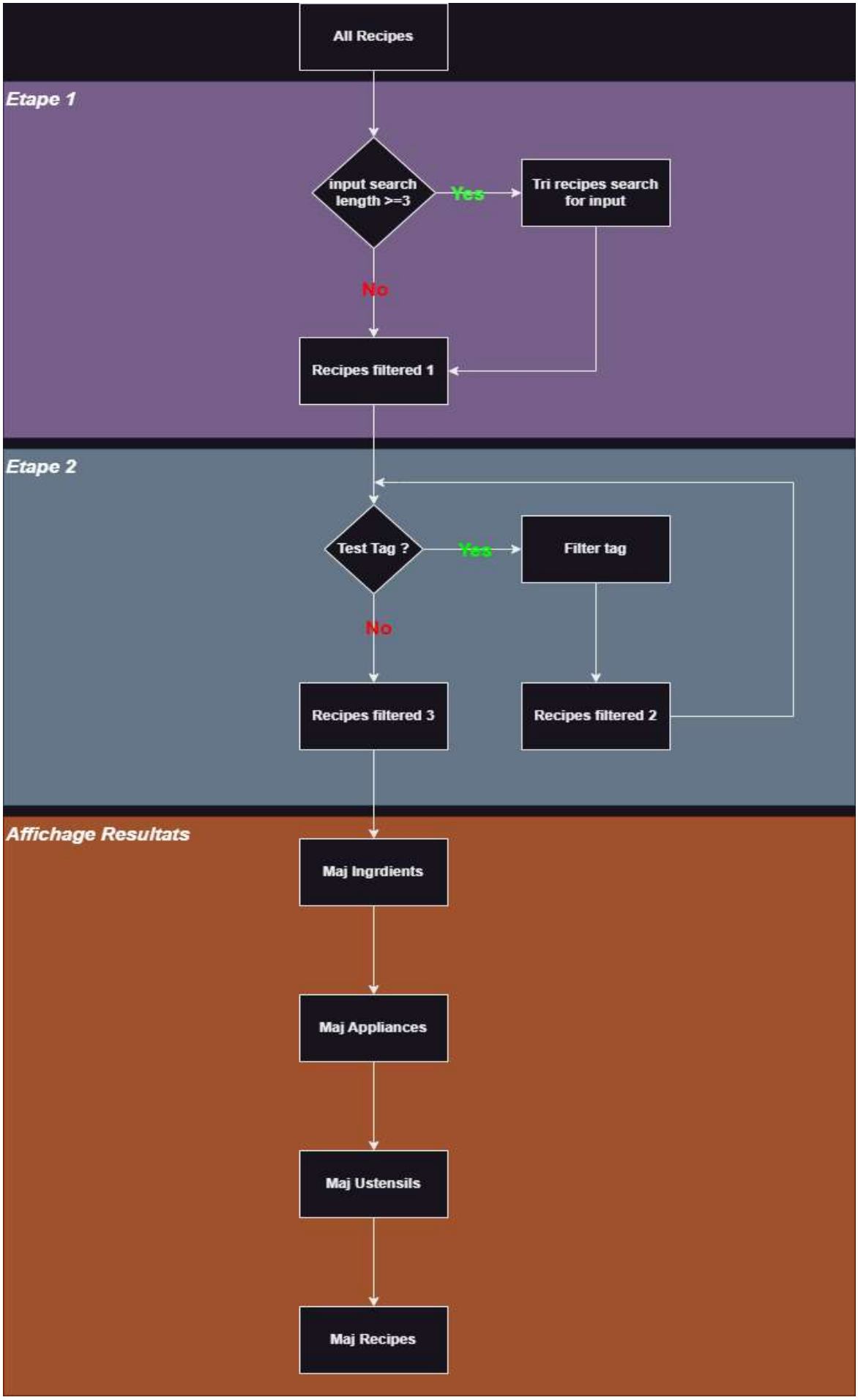
100%

Search V2 (2866331)

95.08%

If you like to donate (Thank you!):

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


All Recipes
Etape 1
input search length >=3
Yes
Tri recipes search for input
No
Recipes filtered 1
Etape 2
Test Tag ?
Yes
Filter tag
No
Recipes filtered 3
Recipes filtered 2
Affichage Resultats
Maj Ingrdients
Maj Appliances
Maj Ustensils
Maj Recipes