

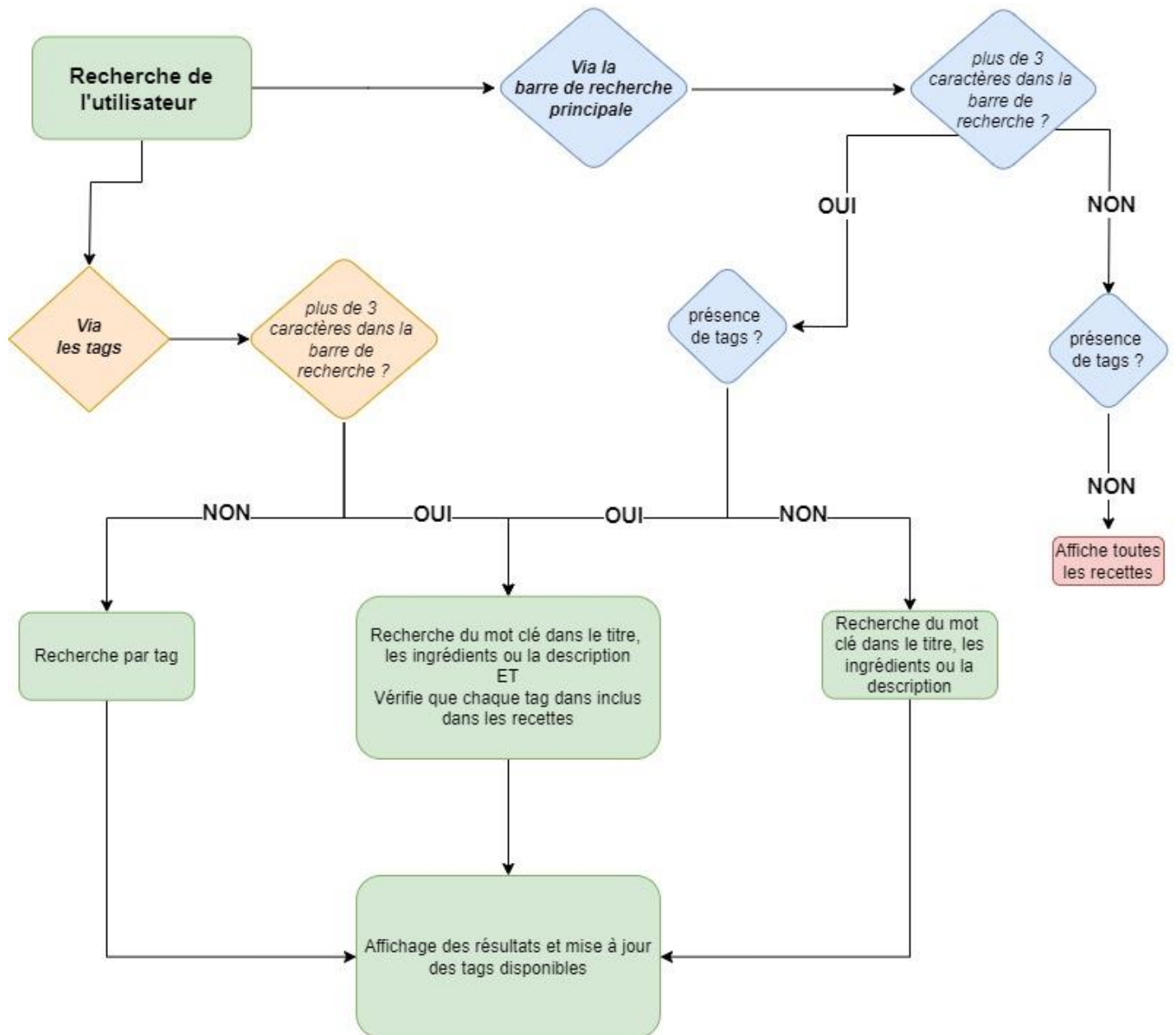
Fiche d'investigation de fonctionnalité

Fonctionnalité : Recherche de recettes	Fonctionnalité #2
<p>Problématique :</p> <p>Permet de trier parmi les recettes disponibles en prenant en compte plusieurs paramètres :</p> <ul style="list-style-type: none">– Un champ de recherche principal (optionnel)– Trois champs de sélection avancés (ingrédients, appareils et ustensiles. Optionnels) <p>La recherche renverra les recettes dont le nom, la description ou les ingrédients contiennent la recherche via la barre de recherche principale, et dont l'appareil, les ingrédients et les ustensiles correspondent aux menus de sélection avancés (tags).</p> <p>Afin de fournir la meilleure expérience utilisateur, j'ai testé deux algorithmes de recherche différent :</p> <ul style="list-style-type: none">– Via la boucle native for ... of– Via la méthode reduce()	

Option 1 : Boucle native for...of	
<p>Avantages :</p> <ul style="list-style-type: none">• Simple à comprendre et à mettre en place• Lisible et maintenable• Très performant si le site comporte beaucoup de recettes (à partir de 200)	<p>Inconvénients :</p> <ul style="list-style-type: none">• Moins performant si pas beaucoup de recettes (moins de 200)

Option 2 : Méthode de l'objet Array reduce	
<p>Avantages :</p> <ul style="list-style-type: none">• Globalement simple à comprendre et à mettre en place• Lisible et maintenable• Performant si le site comporte peu de recettes.	<p>Inconvénients :</p> <ul style="list-style-type: none">• Moins facile à mettre en place et à comprendre pour certains (nouveau de ES6)• Problème de scalabilité

Algorithme :



Comparaison entre for...of et reduce

Sur une base de 50 recettes :

foor of loop vs reduce method v1 - by

compares the **fastest algorithm with a database of 50 recipes**

Setup HTML - click to add setup HTML

Setup JavaScript

```
const data = [
  {
    "id": 1,
    "name": "Limonade de Coco",
    "servings": 1,
    "ingredients": [
      {
        "ingredient": "Lait de coco",
        "quantity": 400,
        "unit": "ml"
      }
    ]
  }
]
```

reduce method

finished

49181181.85 ops/s \pm 9.59%

Fastest

```
checkValues_FilterRecipes = () => {
  const searchBar = document.querySelector('form .search-bar');
  let searchBarValue = '';
  if(searchbar.value.length >= 3) {
    searchBarValue = searchBar.value.toLowerCase().trim();
  }

  this.filteredResult = data.reduce((filteredRecipes, currentRecipe) => {
    let matchIngredient, matchAppliance, matchUstensil;

    if(this.currentTags.ingredients.every(tagIngredient =>
      currentRecipe.ingredients.some(ingredient => ingredient.toLowerCase().includes(tagIngredient.toLow
```

for of loop

finished

41067872.35 ops/s \pm 10.76%

16.5 % slower

```
checkValues_FilterRecipes = () => {
  this.filteredResult = new Set();
  const searchBar = document.querySelector('form .search-bar');
  let searchBarValue = '';
  if(searchbar.value.length >= 3) {
    searchBarValue = searchBar.value.toLowerCase().trim();
  }

  for(let currentRecipe of data) {
    let matchIngredient, matchAppliance, matchUstensil;

    // si une recette contient tous les tag ET le mot clé dans la barre de recherche => matchIngredient = true
```

Résultat : reduce est 16% plus rapide

100 recettes :

foor of loop vs reduce method

v1

- by

compares the fastest algorithm with a database of 100 recipes

Setup HTML - click to add setup HTML

Setup JavaScript

```

        "ingredient": "Sucre glace",
        "quantity": 500,
        "unit": "grammes"
      },
      "time": 60,
      "description": "Préparer la frangipane : Mélanger le sucre la poudre d'amander, le beurre",
      "appliance": "Four",
      "ustensils": ["rouleau à pâtisserie", "fouet"]
    },
  ],
  {
    // si une recette contient tous les tag ET le mot clé dans la barre de recherche => m
  }

```

reduce method

finished

41281793.31 ops/s ± 11.4%

Fastest

```

checkValues_FilterRecipes = () => {
  const searchbar = document.querySelector('form .search-bar');
  let searchbarValue = '';
  if(searchbar.value.length >= 3) {
    searchbarValue = searchbar.value.toLowerCase().trim();
  }

  this.filteredResult = data.reduce((filteredRecipes, currentRecipe) => {
    let matchIngredient, matchAppliance, matchUstensil;

    if(this.currentTags.ingredients.every(tagIngredient =>
      currentRecipe.ingredients.some(ingredient => ingredient.toLowerCase().includes(tagIngredient.toLowerCase())))
      matchIngredient = true;

    if(this.currentTags.appliance === currentRecipe.appliance) matchAppliance = true;

    if(this.currentTags.ustensils.some(ustensil => currentRecipe.ustensils.includes(ustensil))) matchUstensil = true;

    return matchIngredient & matchAppliance & matchUstensil ? [...filteredRecipes, currentRecipe] : filteredRecipes;
  }, []);
}

```

for of loop

finished

46467118.18 ops/s ± 8.27%

Fastest

```

checkValues_FilterRecipes = () => {
  this.filteredResult = new Set();
  const searchbar = document.querySelector('form .search-bar');
  let searchbarValue = '';
  if(searchbar.value.length >= 3) {
    searchbarValue = searchbar.value.toLowerCase().trim();
  }

  for(let currentRecipe of data) {
    let matchIngredient, matchAppliance, matchUstensil;

    // si une recette contient tous les tag ET le mot clé dans la barre de recherche => m
  }
}

```

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

[Wiki](#) | [Report issue](#) | [Become a sponsor!](#)

Inspired by [Benchmark.js](#), [Jsperf.com](#) and [JsFiddle.com](#).

Résultat : les 2 méthodes sont quasiment équivalentes

200 recettes :

foor of loop vs reduce method
v1
- by
-

compares the fastest algorithm with a database of 200 recipes

+ Setup HTML - click to add setup HTML

Setup JavaScript

```
const data = [
  {
    "id": 1,
    "name": "Limonade de Coco",
    "servings": 1,
    "ingredients": [
      {
        "ingredient": "Lait de coco",
        "quantity": 400,
        "unit": "ml"
      }
    ]
  }
]
```

reduce method

finished

37436251.35 ops/s \pm 8.72%
15.41 % slower

```
checkValues_FilterRecipes = () => {
  const searchbar = document.querySelector('form .search-bar');
  let searchbarValue = '';
  if(searchbar.value.length >= 3) {
    searchbarValue = searchbar.value.toLowerCase().trim();
  }

  this.filteredResult = data.reduce((filteredRecipes, currentRecipe) => {
    let matchIngredient, matchAppliance, matchUstensil;

    if(this.currentTags.ingredients.every(tagIngredient =>
      currentRecipe.ingredients.some(ingredient => ingredient.toLowerCase().includes(tagIngredient.toLowerCase()))) {
      matchIngredient = true;
    }

    if(this.currentTags.appliance.includes(currentRecipe.appliance.toLowerCase())) {
      matchAppliance = true;
    }

    if(this.currentTags.ustensil.includes(currentRecipe.ustensil.toLowerCase())) {
      matchUstensil = true;
    }

    if(matchIngredient & matchAppliance & matchUstensil) {
      filteredRecipes.add(currentRecipe);
    }

    return filteredRecipes;
  }, new Set());

  return filteredRecipes;
}
```

for of loop

finished

44255877.92 ops/s \pm 7.11%
Fastest

```
checkValues_FilterRecipes = () => {
  this.filteredResult = new Set();
  const searchbar = document.querySelector('form .search-bar');
  let searchbarValue = '';
  if(searchbar.value.length >= 3) {
    searchbarValue = searchbar.value.toLowerCase().trim();
  }

  for(let currentRecipe of data) {
    let matchIngredient, matchAppliance, matchUstensil;

    // si une recette contient tous les tag ET le mot clé dans la barre de recherche => ma
  }
}
```

+ 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

[Wiki](#) | [Report issue](#) | [Become a sponsor!](#)

Inspired by [Benchmark.js](#), [Jsperf.com](#) and [JsFiddle.com](#).

Résultat : la boucle for est 35% plus rapide

Conclusion :

J'ai choisi de retenir l'option 1 : la boucle native for of. En effet, malgré qu'en l'état actuel du site et de ses 50 recettes la méthode reduce est un poil plus rapide que la boucle for, si la base de données venait à s'agrandir et contenir plus de recettes, il serait beaucoup plus efficace d'utiliser la boucle for que la méthode reduce.