



## Fiche d'investigation de fonctionnalité

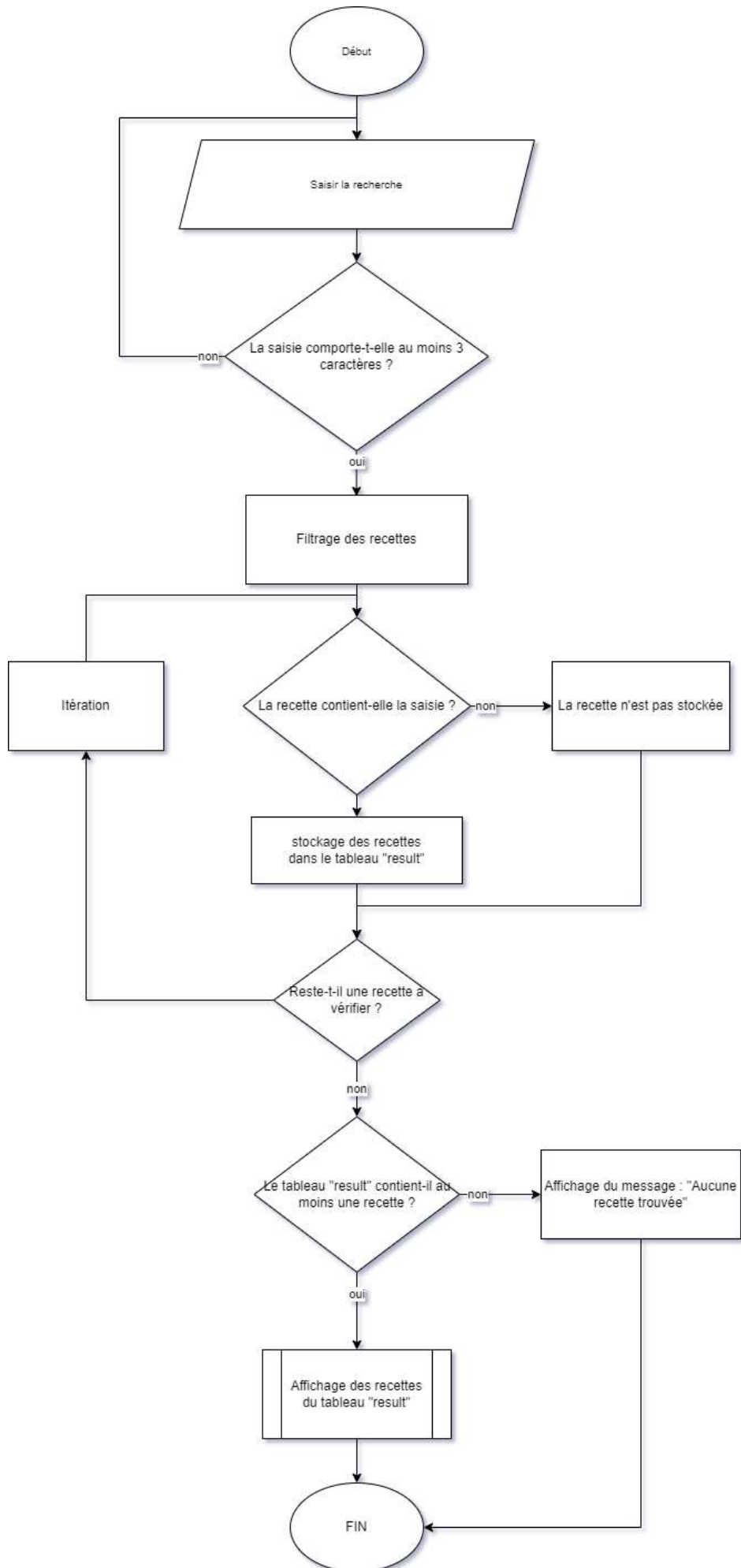
<b>Fonctionnalité :</b> Recherche principale	Fonctionnalité #1
<b>Problématique :</b> Afin de répondre aux besoins des utilisateurs, la recherche doit être la plus rapide possible.	

<b>Option 1 : Programmation fonctionnelle (ES5+) (<u>Annexe 1</u>)</b> <i>Cette option utilise les méthodes de l'objet Array (foreach, filter, map, reduce)</i>	
<b>Avantage(s) :</b> <ul style="list-style-type: none"><li>• Permet de coder de manière plus rapide, claire et concise.</li><li>• Plus modulable (flexible).</li><li>• Moins de bugs et donc plus stable (limite de la dette technique)</li></ul>	<b>Inconvénient(s) :</b> <ul style="list-style-type: none"><li>• Certaines incompatibilités avec des navigateurs.</li></ul>

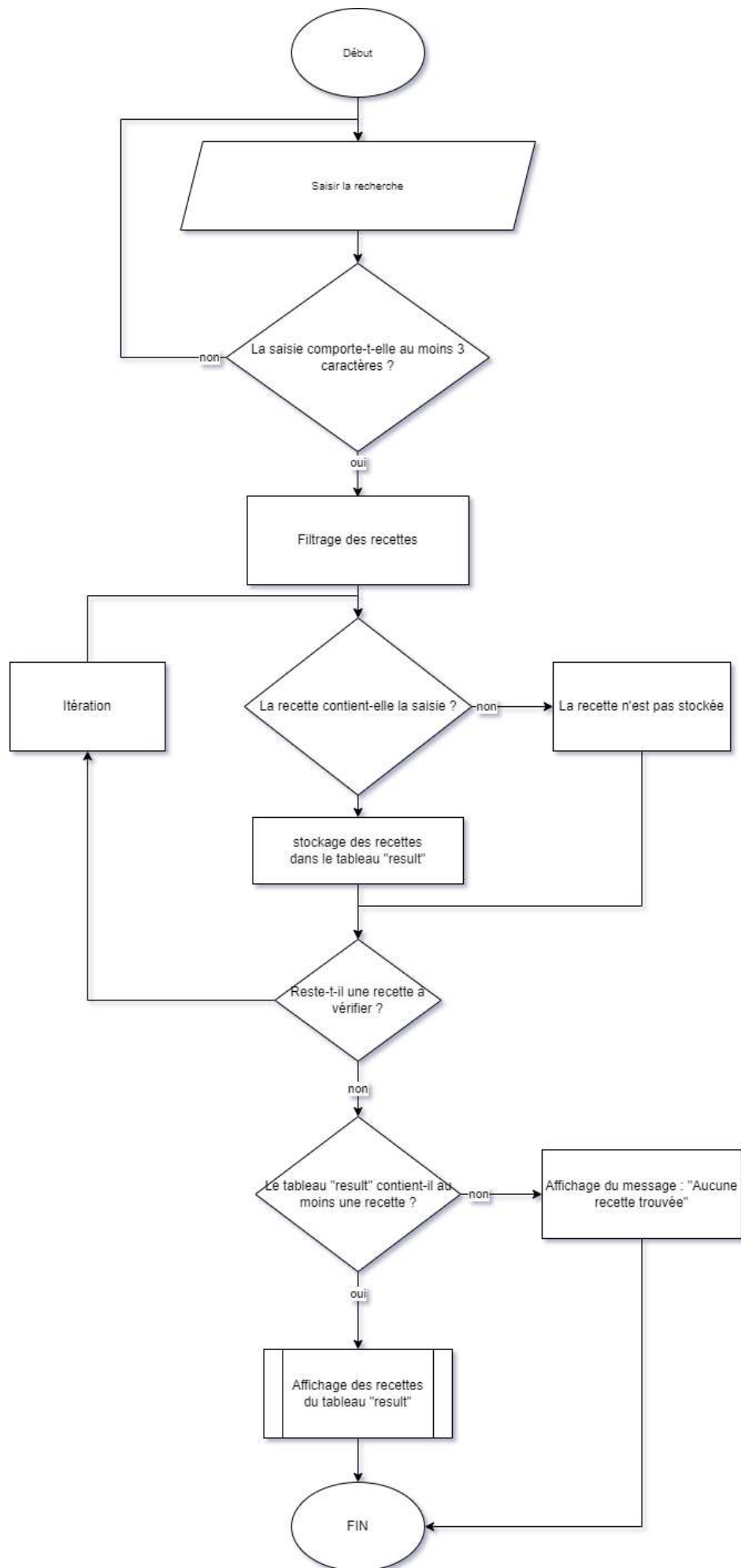
<b>Option 2: Programmation imperative (boucles natives, ES1st) (<u>Annexe 2</u>)</b> <i>Cette option utilise les boucles natives (while, for)</i>	
<b>Avantage(s):</b> <ul style="list-style-type: none"><li>• Compatibilité avec tous les navigateurs</li></ul>	<b>Inconvénient(s) :</b> <ul style="list-style-type: none"><li>• Difficile à comprendre</li><li>• Pas réutilisable / générique</li><li>• Bugs probables</li><li>• Difficile à tester</li></ul>

<b><u>Solution retenue :</u></b> <p>Nous avons retenu l'option#1 et son approche fonctionnelle. Elle permet de gagner du temps sur le développement des solutions. Elle garantit une certaine flexibilité pour l'ajout de futures fonctionnalités. Le code sera plus robuste et stable mais aussi plus facilement maintenable et lisible. Le relevé de performance réalisé à l'aide de « bench.me », nous fait remarquer une différence de performance en faveur de l'option 1.</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Annexe 1 : Programmation fonctionnelle



## Annexe 2 : Boucles natives



## Annexe 3 : JsBench rapport de performances

<https://jsbench.me/3ol1lwqnaf/2>

Chrome :

Investigation performance AlgoV1vsV2 v1 - by alexandre 05/04/2022 0 0

enter test suite description

Setup HTML - click to add setup HTML

Setup JavaScript

```
const recipes = {
  "recipes": [
    {
      "id": 1,
      "name": "Limonade de Coco",
      "image": "assets/img/limonadeCoco.jpg",
      "servings": 1,
      "ingredients": [
        {
          "ingredient": "Lait de coco",
          "quantity": 400,
          "unit": "ml"
        }
      ]
    }
  ]
}
```

Algo1 approche fonctionnelle simple word search (ES5+)

finished

33695.93 ops/s  $\pm$  0.76%

**Fastest**

```
allRecipes.forEach(recipe => recipe.allIngredients = (recipe.ingredients.map((ing) => ing.ingredient)));
let matchingRecipes = allRecipes;
result = matchingRecipes.filter((recipe) =>
  (`${recipe.name} ${recipe.description} ${recipe.allIngredients}`).includes(inputValue)
);
console.log(result.length);
};
filterRecipes();
//Même algorithme que le n°1 sans le RegExp
```

Algo2 approche imperative simple word search (ES1st)

finished

21722.28 ops/s  $\pm$  0.93%

35.53 % slower

```
var inputValue = ["coco"];
var result = [];
var resultFiltered = [];

const allRecipes = recipes.recipes;

//indexOf n'existe que dans le prototype de String
//création d'une fonction pour array
function indexOf(array, obj, start) {
  for (let i = start || 0, j = array.length; i < j; i++) {
    if (array[i] === obj) {
      return i;
    }
  }
}
```

Firefox :

Investigation performance AlgoV1vsV2 v2 06/04/2022 by alexandre 0 0

enter test suite description

Setup HTML - click to add setup HTML

Setup JavaScript

```
    quantity: 400,
    unit: "grammes"
  },
  time: 40,
  description: "Préparer la frangipane : Mélanger le sucre la poudre d'amander, le beurre et les oeufs. Étaler la pâte feuilletée. Mettre au four 30 minutes",
  appliance: "Four",
  ustensils: ["rouleau à pâtisserie", "fouet"]
}
];
};
```

Algo1 approche fonctionnelle simple word search (ES5+)

finished

13292.66 ops/s  $\pm$  1.23%

**Fastest**

```
const {recipes:allRecipes} = recipes

//FILTER RECIPES FUNCTION
const filterRecipes = () => {
  allRecipes.forEach(recipe => recipe.allIngredients = (recipe.ingredients.map((ing) => ing.ingredient)));
  let matchingRecipes = allRecipes;
  result = matchingRecipes.filter((recipe) =>
    (`${recipe.name} ${recipe.description} ${recipe.allIngredients}`).includes(inputValue)
  );
}
```

Algo2 approche imperative simple word search (ES1st)

finished

9628.04 ops/s  $\pm$  2.08%

27.57 % slower

```
var inputValue = ["coco"];
var result = [];
var resultFiltered = [];

const allRecipes = recipes.recipes;

//indexOf n'existe que dans le prototype de String
//création d'une fonction pour array
function indexOf(array, obj, start) {
  for (let i = start || 0, j = array.length; i < j; i++) {
    if (array[i] === obj) {
      return i;
    }
  }
}
```

**Tests réalisés sur Vs Code :**

**Temps moyen pour 10.000 exécutions simultanées.**

*Algo1:* **253ms**

|

*Algo2:* **1.635s**