

Fiche d’investigation fonctionnalité #1

**Fiche d’investigation de fonctionnalité**

**Solution retenue :**

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.

**Inconvénient(s) :**

* Difficile à comprendre
* Pas réutilisable / générique
* Bugs probables
* Difficile à tester

**Option 2: Programmation imperative (boucles natives, ES1st) (*Annexe 2*)**

*Cette option utilise les boucles natives (while, for)*

**Avantage(s):**

* Compatibilité avec tous les navigateurs

**Inconvénient(s) :**

* Certaines incompatibilités avec des navigateurs.

**Avantage(s) :**

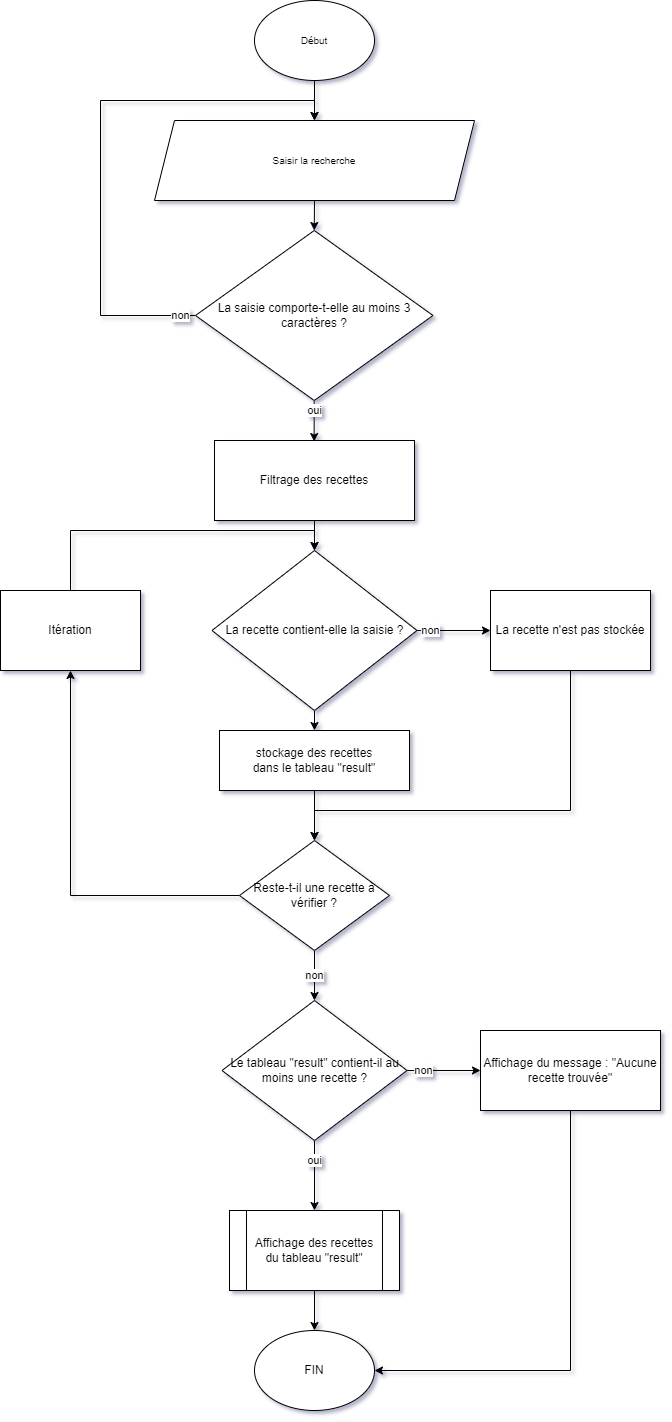
* *Permet de coder de manière plus rapide, claire et concise.*
* *Plus modulable (flexible).*
* *Moins de bugs et donc plus stable (limite de la dette technique)*

**Fonctionnalité :** Recherche principale Fonctionnalité #1

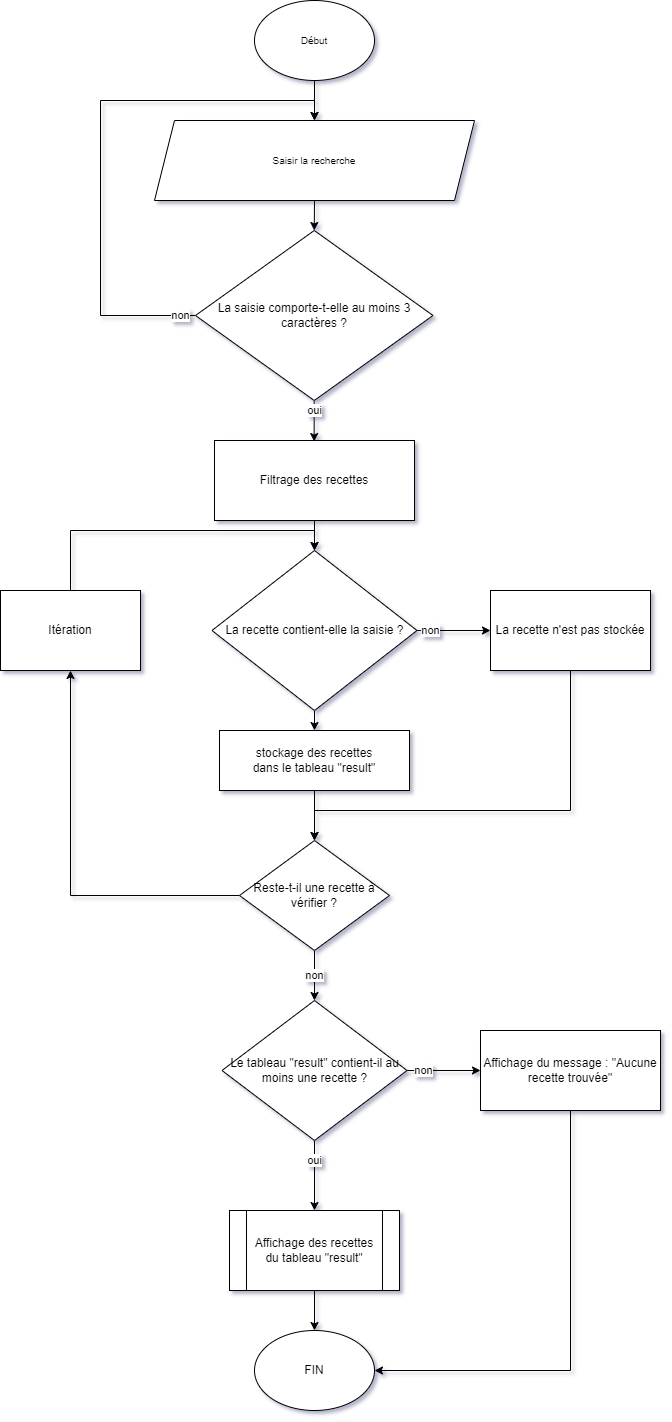
**Problématique :** Afin de répondre aux besoins des utilisateurs, la recherche doit être la plus rapide possible.

**Option 1 : Programmation fonctionnelle (ES5+)  (*Annexe 1*)**

*Cette option utilise les méthodes de l’objet Array (foreach, filter, map, reduce)*

**Annexe 1 : Programmation fonctionnelle**

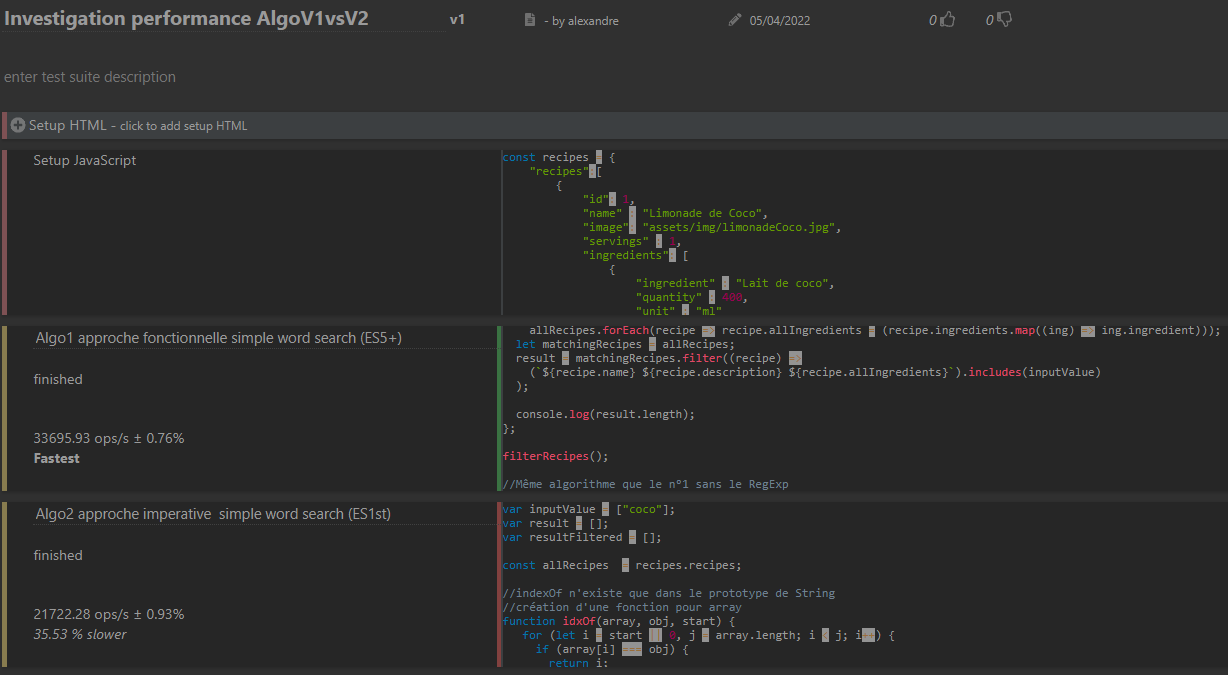
**Annexe 2 : Boucles natives**

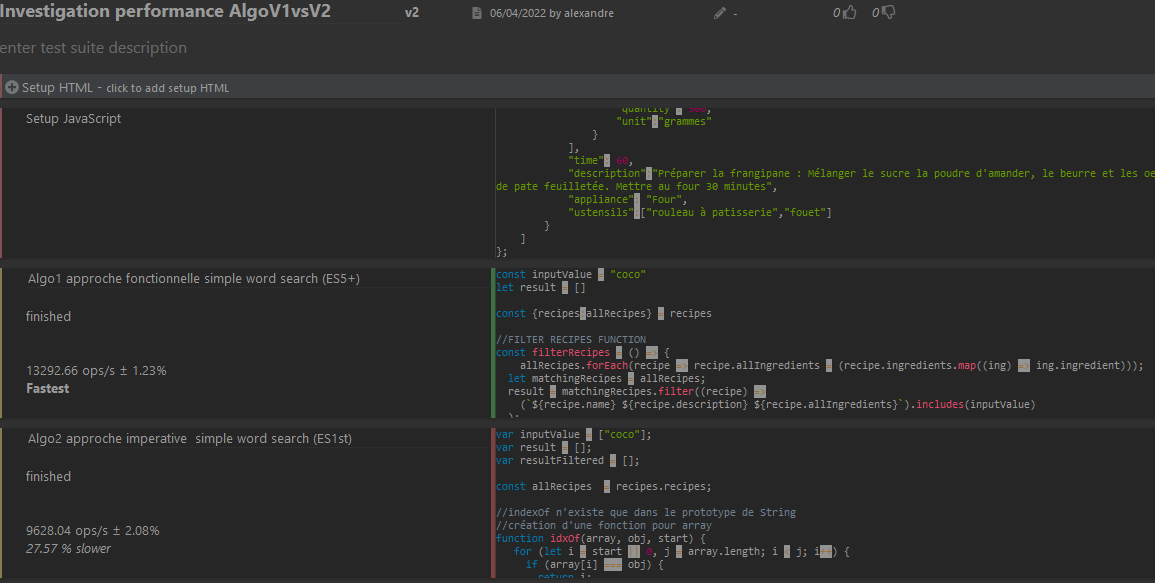
****

**Annexe 3 : JsBench rapport de performances**

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

Chrome :



Firefox :

**Tests réalisés sur Vs Code :**

**Temps moyen pour 10.000 exécutions simultanées.**

*Algo1*: **253ms** | *Algo2*: **1.635s**