Les petits plats

Fiche d'investigation fonctionnalité #1

Fiche d'investigation de fonctionnalité

Fonctionnalité : Recherche principale	**Fonctionnalité #1**
Problématique : Afin de pouvoir retenir un maximum d'utilisateurs, nous cherchons à avoir un résultat de recherche quasi instantané.	

Option 1 : Algo V1 (cf algorigramme)
Dans cette option, la recherche est effectuée à l'aide d'un .filter et d'un .includes directement sur l'objet (array recipes)

Avantages	**Inconvénients**
⊕ Code "basique" -> simple recherche sur chacun des "éléments" de l'objet (name, ingredients,ustensils, description, appliance) ⊕ Si "l'objet" change il n'y a que la partie recherche à modifier	⊖ La recherche se fait par itération (on vérifie que la valeur dans l'input n'est pas présente dans le name, puis dans la description, les ingrédients, les ustensils et les appliances… et on renvoie les recettes qui comporte l'input (si il y'en a). Cette méthode peut s'avérer chronophage.

Nombre de caractères pour déclencher la recherche : 3

Option 2 : Algo V2 (cf algorigramme)
Dans cette option, on effectue un prétraitement ("nettoyage") du tableau/objet (recipes). Le nettoyage est déclenché à l'arrivée sur le site pour optimiser le temps de la recherche.
Le nettoyage : suppression de certains mots, plus de casses majuscule/minuscule, plus de ponctuation, plus de chiffres et création d'un string pour éviter l'itération sur les ingrédients.
Pour l'affichage du résultat, obligation d'effectuer une recherche inversée sur l'objet initial.
Tout comme pour la V1 la recherche se fait via un .filter et un .includes par itération sur les différents éléments de l'objet après nettoyage.

Avantages	**Inconvénients**
⊕ Réduction des données sur lesquelles la recherche est effectuée. ⊕ Préchargement	⊖ Devoir faire une recherche inversée pour l'affichage du résultat ⊖ La recherche se fait par itération (on vérifie que la valeur dans l'input n'est pas présente dans le name, puis dans la description, les ingrédients, les ustensils et les appliances… et on renvoie les recettes qui comporte l'input (si il y'en a). Cette méthode peut s'avérer chronophage.

Nombre de caractères pour déclencher la recherche : 3

Option 3 : Algo V3 (cf algorigramme)

Dans cette option, on effectue un prétraitement (“nettoyage” plus poussé que dans la V2) du tableau/objet (recipes).
Le nettoyage est déclenché à l’arrivée sur le site pour optimiser le temps de la recherche.
Le nettoyage : suppression de certains mots, tout en majuscule, plus de ponctuation, plus de chiffres, pas de doublons de mots.
Création d’un objet suite au nettoyage : chaque mot “unique” renvoie le ou les id de sa présence dans la/les recette(s), les mots sont dans l’ordre alphabétique.
La recherche dichotomique est effectuée (on compare à l'élément médian de l’objet puis si besoin sur l’élément médian de la première ou seconde partie..etc.
Lorsqu’il y a un résultat une vérification est effectuée sur l’élément précédent ainsi que sur tous les éléments suivants tant que ces éléments correspondent aussi.
Pour l’affichage du résultat, obligation d’effectuer une recherche inversée sur l’objet initial.

Avantages	**Inconvénients**
⊕ Grande réduction des données sur lesquelles la recherche est effectuée. (497 mots pour l’objet recipes actuel) ⊕ Préchargement ⊕ La recherche dichotomique permet de réduire grandement le temps de recherche (pas d’itérations)	⊖ Devoir faire une recherche inversée pour l’affichage du résultat

Nombre de caractères pour déclencher la recherche : 3

Solution retenue :

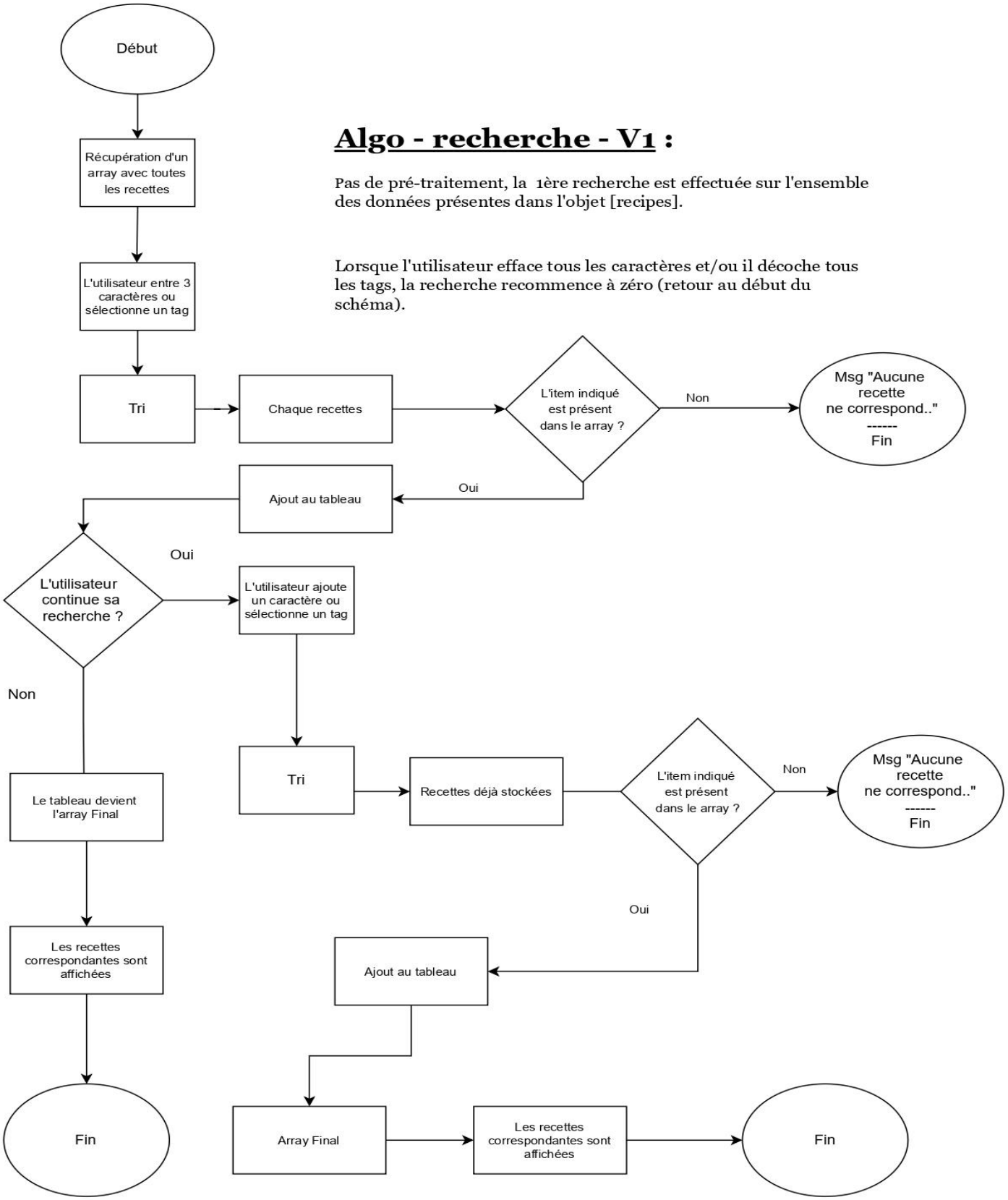
Nous avons donc retenu la V3 qui, bien qu’elle demande un certain travail pour la préparation de l’objet, reste quand même l’approche la plus rapide pour effectuer la recherche principale.

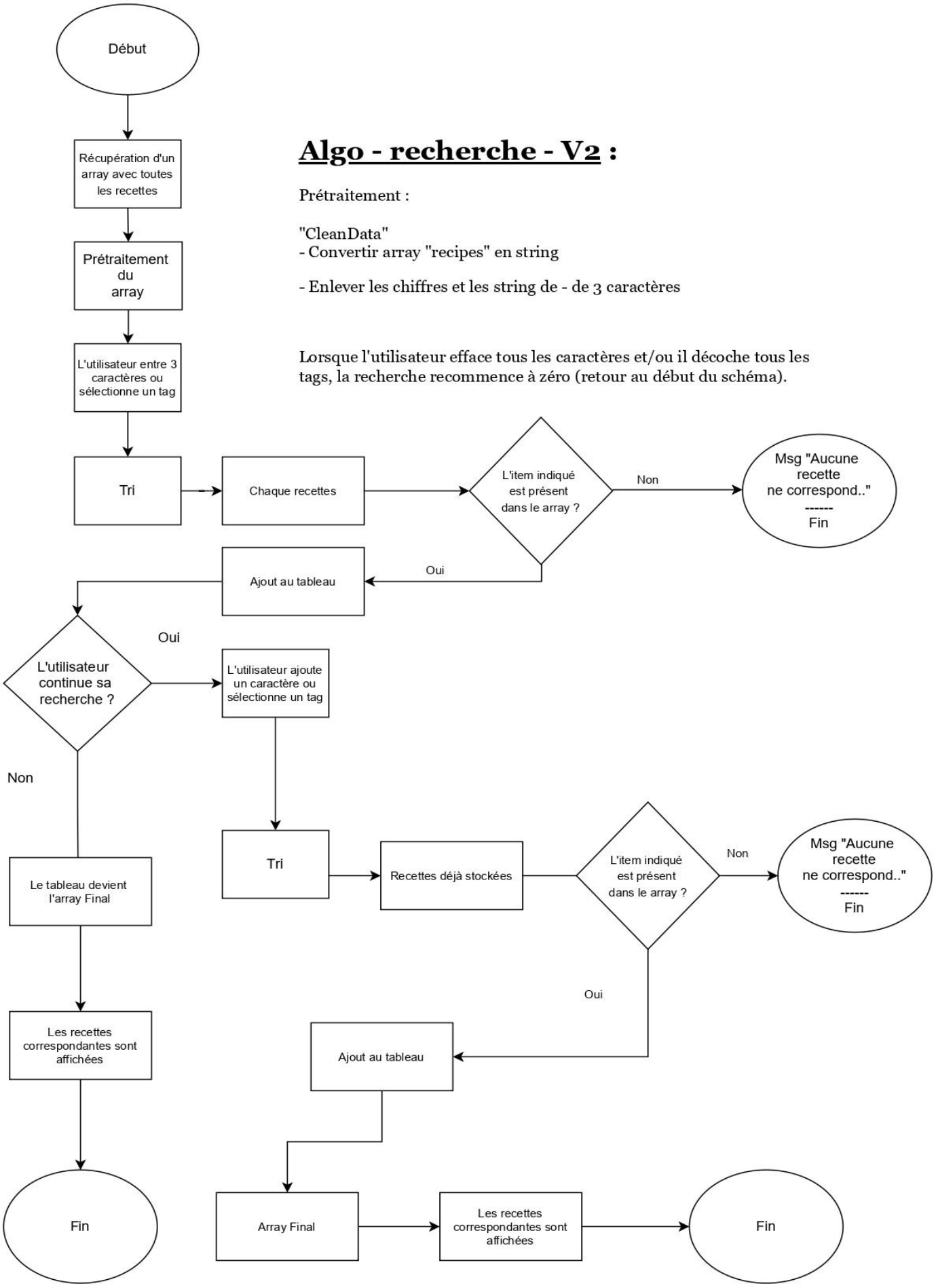
Annexes :

Algo - recherche - V1 :

Pas de pré-traitement, la 1ère recherche est effectuée sur l'ensemble des données présentes dans l'objet [recipes].

Lorsque l'utilisateur efface tous les caractères et/ou il décoche tous les tags, la recherche recommence à zéro (retour au début du schéma).





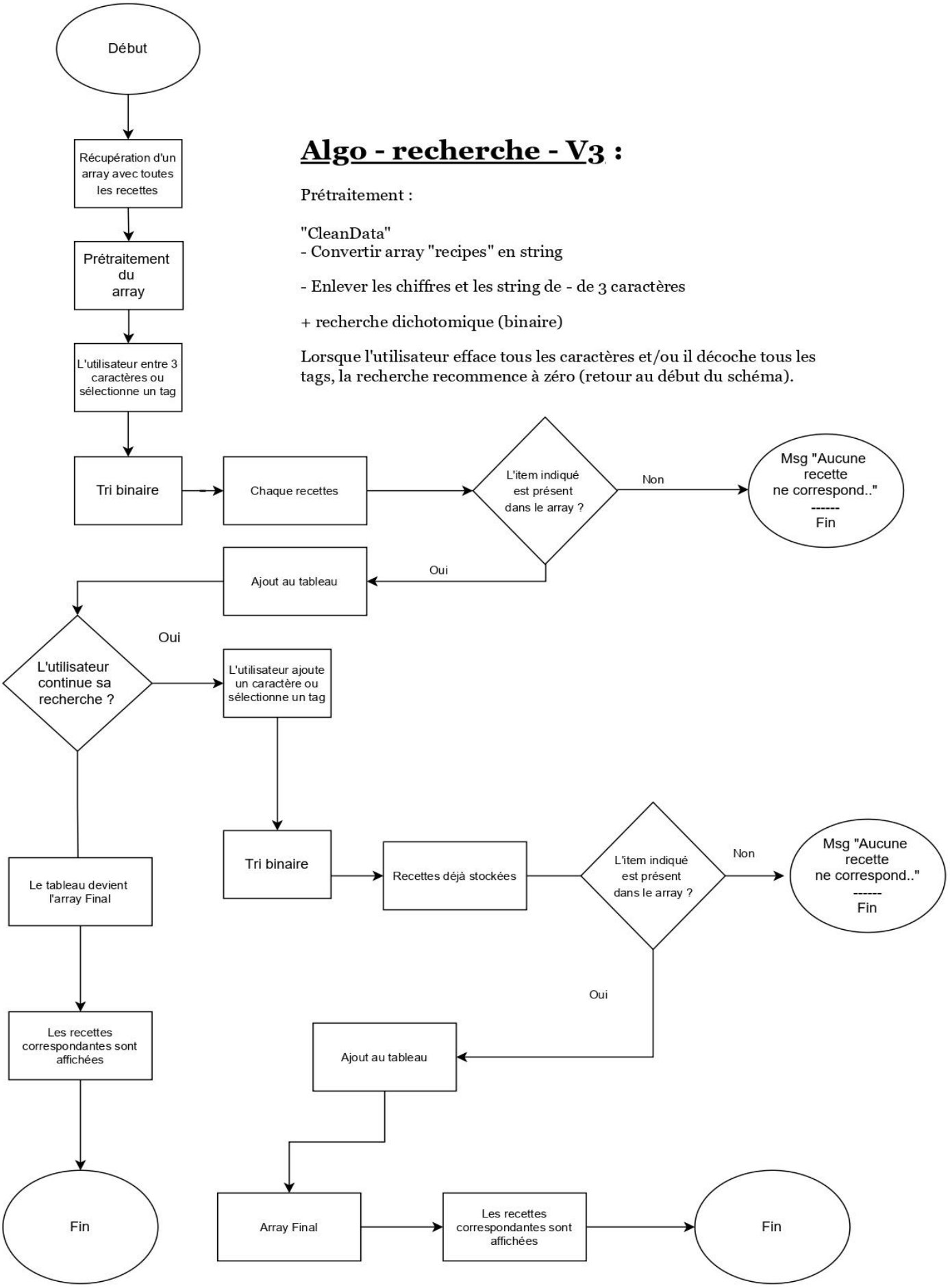
Algo - recherche - V2 :

Prétraitement :

"CleanData"
- Convertir array "recipes" en string

- Enlever les chiffres et les string de - de 3 caractères

Lorsque l'utilisateur efface tous les caractères et/ou il décoche tous les tags, la recherche recommence à zéro (retour au début du schéma).



Algo - recherche - V3 :

Prétraitement :

"CleanData"
- Convertir array "recipes" en string

- Enlever les chiffres et les string de - de 3 caractères

+ recherche dichotomique (binaire)

Lorsque l'utilisateur efface tous les caractères et/ou il décoche tous les tags, la recherche recommence à zéro (retour au début du schéma).

ANALYSE DES DIFFÉRENTS ALGORITHMES

(à l'aide de JSBench)

Voici mon analyse des tests effectués à l'aide de JSBench.me (https://jsbench.me/) :

Test effectué sur le mot "coco" :

V1	8116.47 ops/s	74.18% slower	2nd
V2	7923.92 ops/s	74.79% slower	3rd
V3	31436.13 ops/s	Fastest	1st

Test effectué sur le terme "tart" (qui permet de cibler les mots tarte, tartelette, tartiflette..) :

V1	15416.78 ops/s	54.25% slower	2nd
V2	9120.4 ops/s	72.93% slower	3rd
V3	33694.32 ops/s	Fastest	1st

Test effectué sur le mot "test" (qui permet d'avoir un résultat de 0 correspondances) :

V1	13976.64 ops/s	88.86% slower	2nd
V2	7667.14 ops/s	93.89% slower	3rd
V3	125503.87 ops/s	Fastest	1st

En ordre de rapidité :

- ➢ V3 - recherche dichotomique
- ➢ V1 - recherche simple
- ➢ V2 - recherche avec tableau nettoyé

La V1 est plus rapide que la V2 mais moins rapide que la V3.
Cela s'explique par le traitement d'affichage de la/les recette(s) trouvée(s) de la V2 qui ne peuvent être affichée(s) que par le tableau recipes original néanmoins la V3 qui elle aussi à besoin d'une recherche inversée reste quand même plus rapide grâce à l'aide de sa recherche dichotomique, elle obtient 3 fois plus vite le/les résultats.

Dans chacun des tests effectués la V3 obtient les meilleurs résultats en termes de rapidité et d'opérations effectuées à la seconde, c'est pour cela que je choisis cette version qui pour moi est la meilleure. De plus, la recherche binaire est préconisée sur les structures de données importantes (ce qui à terme devrait être le cas concernant le nombre de recettes proposées).

Ces résultats sont basés sur le tableau recipes contenant 50 recipes, en fonction de la taille de ce tableau (nombre de recettes, nombre d'éléments…) les résultats peuvent être amenés à changer (concernant la V1 et V2).

Les résultats du test peuvent être vérifiés manuellement sur le site (https://jsbench.me/) à l'aide du fichier code-JSBench.txt fourni (pour cela il suffit de faire un copier-coller des différentes parties dans les zones de JSBench dédiée (setup js, algo 1, algo 2 et algo 3) et de modifier la valeur d'inputValue présente dans le setupJS.