Fiche d'investigation de fonctionnalité

Fonctionnalité : Rechercher une recette de cuisine.

Problématique : Afin de pouvoir retenir un maximum d'utilisateurs, nous cherchons à avoir une séquence de recherche de recette la plus fluide possible.

Description de l'algorithme :

Les données d'entrée de la recherche de recette sont de deux types :

- Premièrement la recherche peut s'effectuer par mots-clés dans la barre de recherche principale,
- Deuxièmement elle peut s'effectuer à l'aide de tags (ingrédients, appareils ou ustensiles).

L'algorithme est pensé comme un système d'entonnoir où la liste de recettes disponibles est de plus en plus restreinte (liste de recettes filtrée) à mesure que les critères se précisent. Ainsi lorsque qu'on ajoute un critère (un caractère supplémentaire au mot-clé ou un tag supplémentaire) l'action de filtre ne s'effectue que sur la liste filtrée en cours et non sur la liste d'origine qui contient toutes les recettes de la base de données. Dans le cas où un critère est modifié et non ajouté (suppression/modification d'un caractère du mot-clé ou suppression d'un tag) l'opération de filtrage est effectuée sur une liste de recettes réinitialisée contenant l'ensemble des recettes de la base de données.

Les deux algorithmes créés fonctionnent selon la même succession d'étapes (algorigramme) présentée Figure 1.

La différence entre ces deux algorithmes concerne le traitement des tableaux lors de la recherche par mots-clés. En effet cette recherche fait intervenir plusieurs opérations sur des tableaux (cf. Figure 1) :

- Filtrage de la liste de recettes,
- Recherche de mot-clé inclus au sein de listes d'ingrédients,
- Mise à jour de listes de tags,
- Tri alphabétique de listes de tags.

Dans le cas du premier algorithme (array methods) les boucles sur les tableaux sont effectuées à l'aide des méthodes de l'objet « Array » : « filter », « foreach » et « sort ».

Dans le cas du deuxième algorithme (native loops) les boucles sur les tableaux sont effectuées à l'aide de la boucle native javascript : « for ».

Choix de l'algorithme le plus performant :

Il est possible de réaliser des tests de performance d'algorithme basés sur leur fréquence d'exécution à l'aide de la librairie Benchmarkjs.

Afin de déterminer quelle version de l'algorithme est la plus performante des tests sont réalisés sur les quatre séquences décrites dans le paragraphe précédent. Les résultats obtenus sont les suivants :

1. Filtrage de la liste de recettes :

```
// arrayMethod x 20,338,207 ops/sec ±0.44% (96 runs sampled)
// nativeLoops x 4,139,581 ops/sec ±0.33% (96 runs sampled)
// The fastest option is arrayMethod (x 4.91)
```

2. Recherche de mot-clé inclus au sein de listes d'ingrédients :

```
// arrayMethod x 303,196 ops/sec ±0.52% (93 runs sampled)
// nativeLoops x 413,893 ops/sec ±0.42% (92 runs sampled)
// The fastest option is nativeLoops (x 1.37)
```

3. Mise à jour de listes de tags :

```
// arrayMethod x 2,821,343 ops/sec ±0.62% (95 runs sampled)
// nativeLoops x 2,398,956 ops/sec ±0.48% (93 runs sampled)
// The fastest option is arrayMethod (x 1.18)
```

4. Tri alphabétique de listes de tags :

```
// arrayMethod x 390,908 ops/sec ±0.47% (94 runs sampled)
// nativeLoops x 237,854 ops/sec ±0.66% (94 runs sampled)
// The fastest option is arrayMethod (x 1.64)
```

Hormis la 2ème séquence où l'algorithme « native loops » présente une fréquence 1,37 fois plus élevée, l'algorithme « array methods » présente toujours une fréquence plus élevée (de x1,18 à x4,91).

Ainsi l'algorithme « array methods » est jugé plus performant et est conservé pour ce projet.

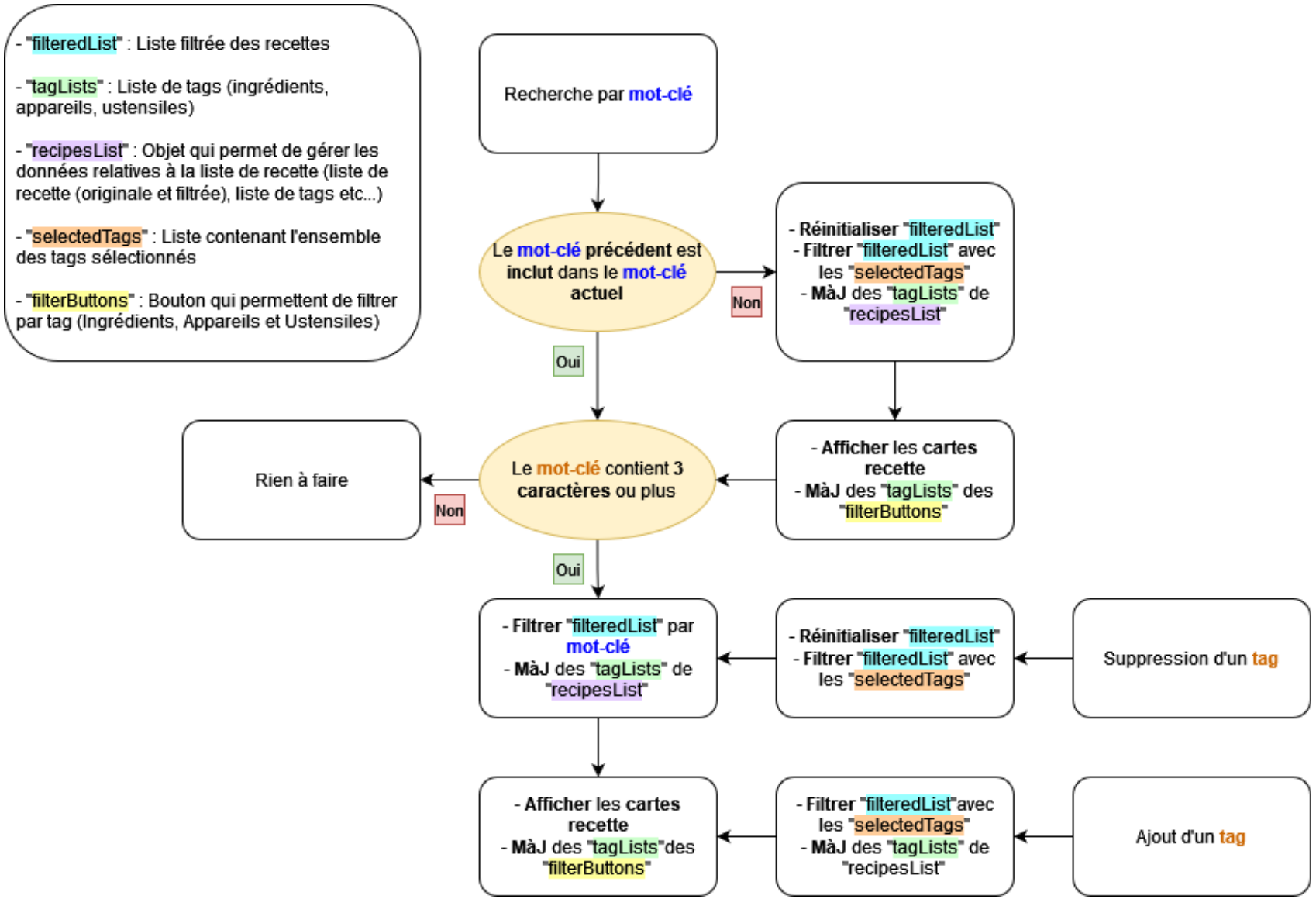

Figure 1. Algorigramme de recherche de recette