

Fiche d'investigation de fonctionnalité

| Fonctionnalité : Barre de recherche des recettes | Fonctionnalité |
|---|----------------|
| Problématique : Permettre d'avoir une barre de recherche des recettes performant utilisant différents filtres, en fonction : du nom, de la description et des ingrédients. | |

| | |
|--|--|
| Option 1 : Fonction de filtrage (filter()) - Figure 1 La fonction de filtrage « filter() » est utilisée pour créer un nouveau tableau qui contient un sous-ensemble des éléments d'un tableau d'origine en fonction de condition(s) donnée(s). Cette fonction prend en entrée un tableau (nos recettes) et une fonction de filtre (nos filtres ingrédients, ustensiles et appareils), qui teste chaque élément du tableau une seul fois (ce qui en fait une méthode efficace). Si la fonction de filtre retourne « true » pour un élément donné (une recette), cet élément est inclus dans le nouveau tableau (nos recettes filtrées). Sinon, il est ignoré. | |
| Avantages <ul style="list-style-type: none"> • Efficace en performance, car elle parcourt les éléments du tableau donné qu'une seule fois. • Facile à utiliser pour les cas de filtrage simple. • Retourne un nouveau tableau (conserve l'original) • Code plus lisible et maintenable. | Inconvénients <ul style="list-style-type: none"> • Peut être utilisé que pour des filtrages simples. • Il n'est pas possible de modifier les éléments du tableau d'origine. |
| Conclusion Approche pouvant être efficace, facile à utiliser, et maintenable. Mais n'est viable que pour des filtres simples. | |

| | |
|---|--|
| Option 2 : Recherche linéaire La recherche linéaire est une méthode permettant de trouver un élément spécifique dans un tableau en parcourant chaque élément du tableau (plusieurs fois si il le faut), jusqu'à ce que l'élément recherché soit trouvé. Cette méthode se base sur des boucles native « for » qui pourra renvoyer un tableau avec les recettes correspondant à la recherche. | |
| Avantages <ul style="list-style-type: none"> • Simple à mettre en place. • Flexible et peut être utilisé pour des cas de recherche plus complexes | Inconvénients <ul style="list-style-type: none"> • Généralement moins efficace en termes de performance car il parcourt les éléments plusieurs fois. • Il peut carrément être inefficace pour des tableaux de grande taille (avec énormément d'éléments) • Lisibilité du code complexe (plusieurs boucles imbriqués) |
| Conclusion : Approche pouvant être efficace en cas de recherche avec beaucoup de complexités, mais rapidement limité si il y a beaucoup d'éléments à analyser. | |

| |
|---|
| Solution retenue : Fonction de filtrage (filter()) J'opterai pour l'option de fonction de filtrage (avec filter()) pour ces avantages de simplicité et de clarté du code. De plus, comme expliqué (dans Figure 3), la quantité de recette voué à être supérieur à 50, privilégiera davantage cette option pour sa performance, sa rapidité. A noter également que cette option sera plus facilement maintenable en cas de modification de ce système de recherche. (donc plus modulable en fonction des besoins futurs) |
|---|

Annexes

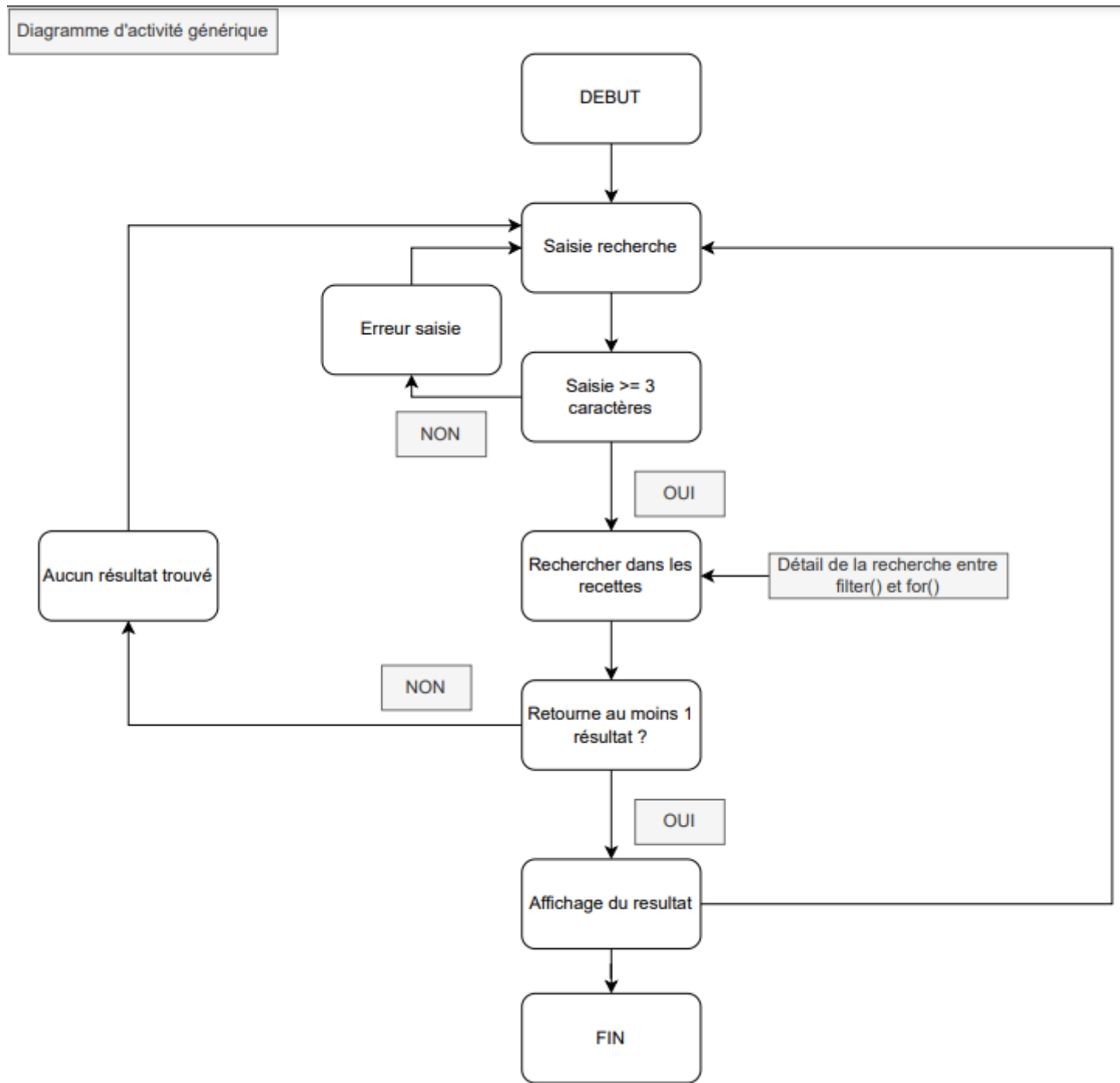


Diagramme d'activité générique

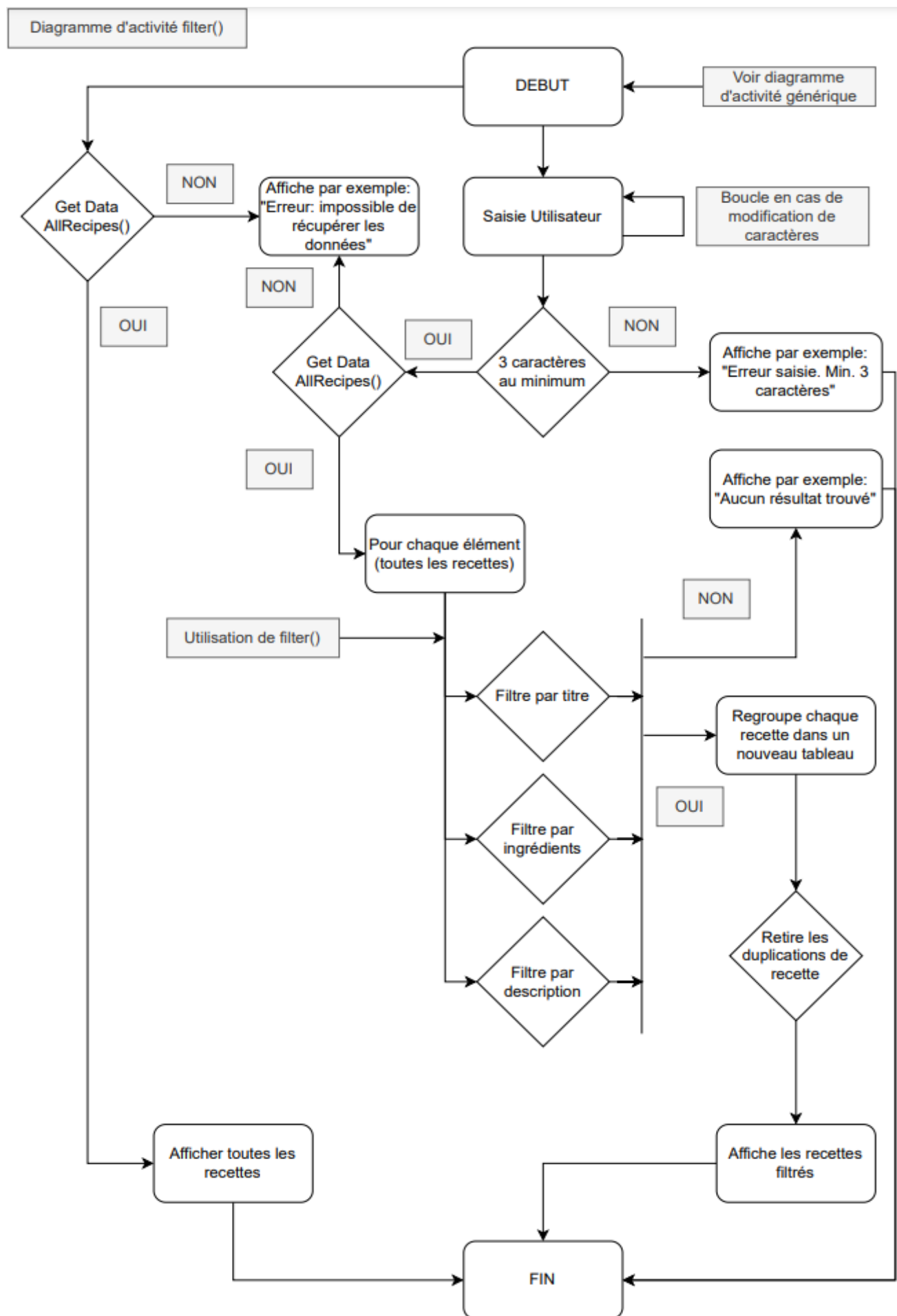


Figure 1 - Diagramme d'activité avec filter()

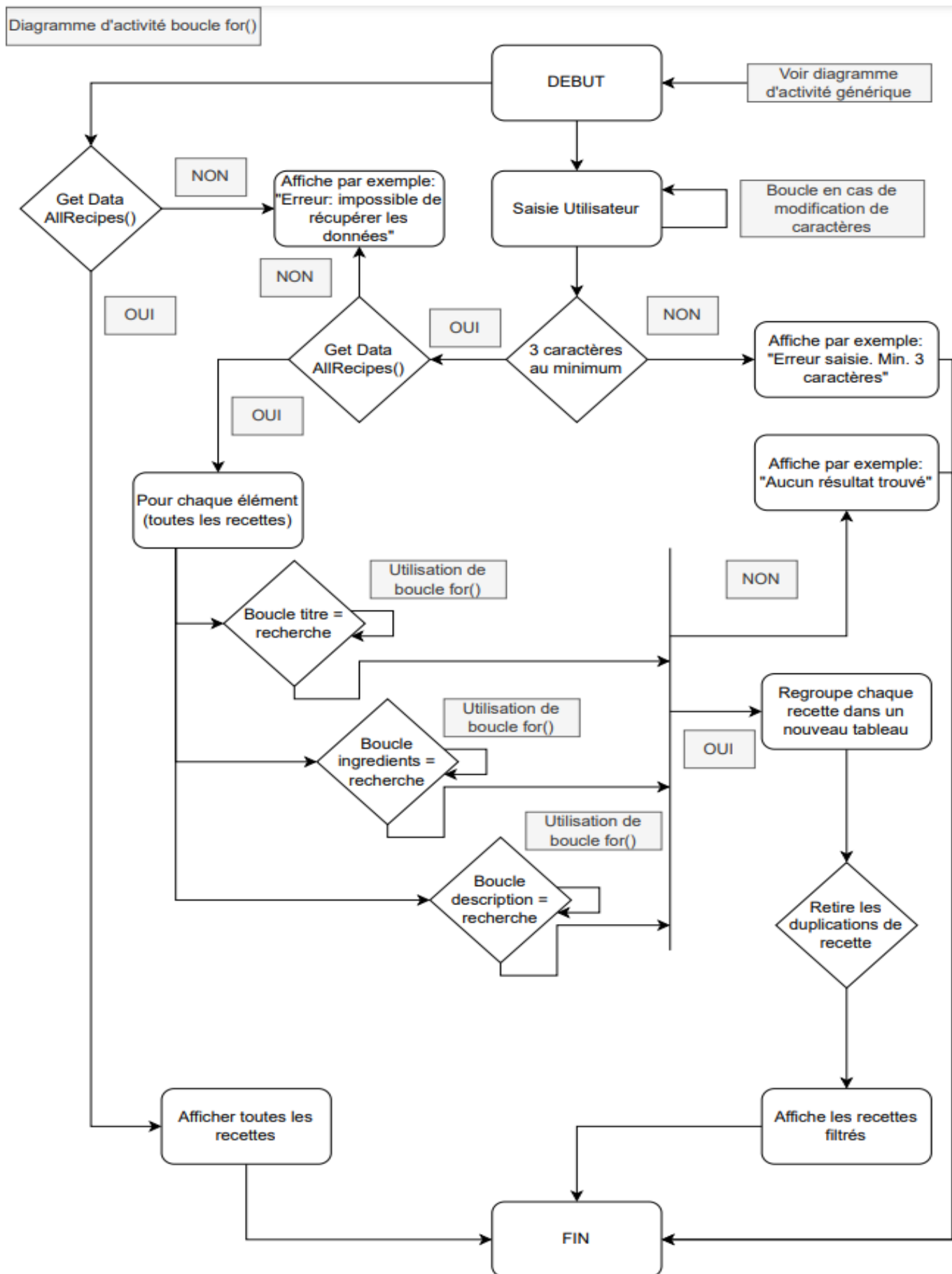
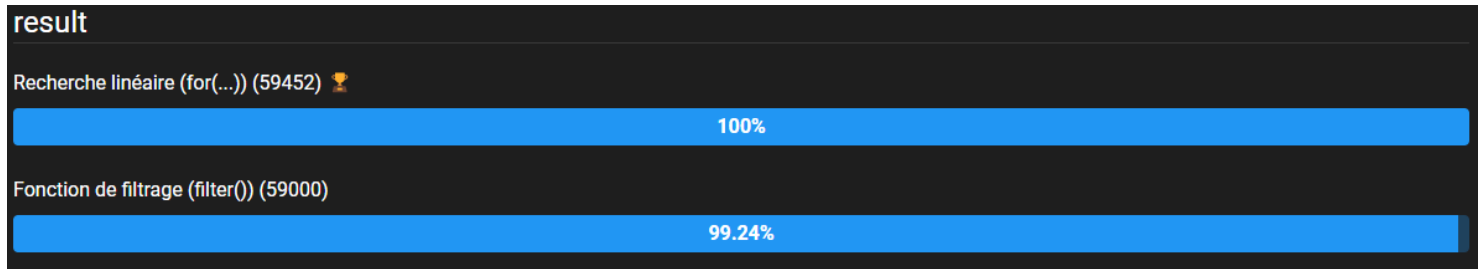


Figure 2 : Diagramme d'activité avec boucles « for »

Résultat de ce test accessible ici : <https://jsben.ch/d0e7D>



A noter que l'outil n'est pas parfait. Si on relance le test (sans pour autant changer les conditions d'algorithme), le résultat ne sera jamais le même. Cela est dû à la manière dont sont calculés les performances. En fonction du navigateur, le résultat ne sera pas non plus le même.

De plus le code utilisé pour le test a dû être adapté pour que le test fonctionne. En effet, celui-ci ne prend en compte que les normes ES6 au maximum (donc pas les nouvelles normes, tels que ES9 pour les map « ... » etc...) se qui influence forcément l'un plutôt que l'autre.

Dernière point, la base de données à dû être diminué à 40 échantillons au lieu de 50 car l'outil a une limite de caractères. Mais il faut savoir que le calcul de performance pour ce genre d'algorithme dépend beaucoup du nombre d'échantillon à analyser. Plus il y aura d'échantillon, plus la méthode par fonction de filtrage sera privilégiée.

Figure 3 : Résultat comparaison algorithme