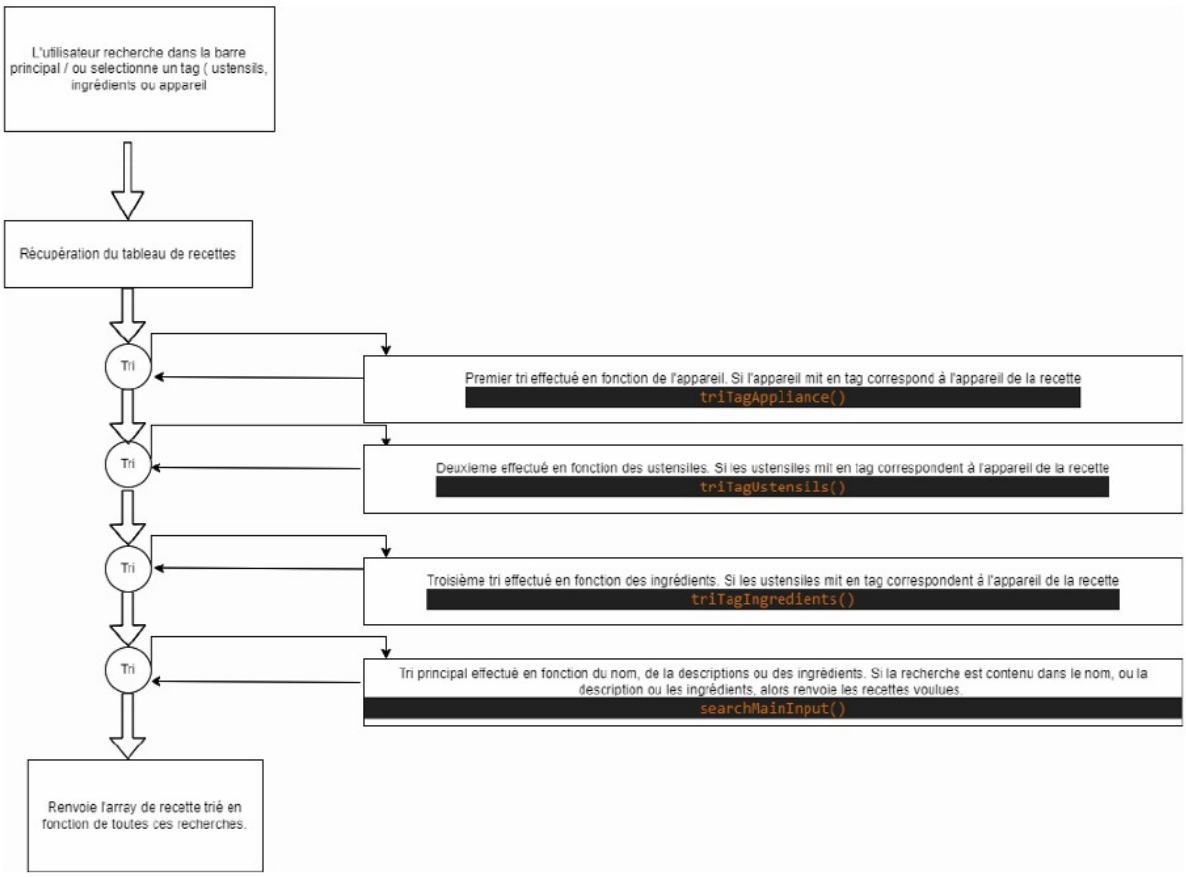
Fiche d'investigation de fonctionnalité

Fonctionnalité de recherche	Fonctionnalité 1
Problématique : Permet de trier les recettes en prenant plusieurs paramètres en comptes : 1) Un champ de recherche principal 2) 3 Champs de selections (ustensiles, ingrédients, appareil) Afin de chercher à fournir le meilleur service à l'utilisateur , j'ai souhaité tester deux implémentations de l'algorithmem de recherche de recette au sein de l'application Web : « Les petits Plats »	

Option 1 : Avec des boucles natives for/ while :	
Avantages : 1) Simple à mettre en place 2) Rapidité d'execution lorsque les listes parcourues sont courtes.	Inconvénients : 1) Lenteur d'éxécution quand la base de donnée peut grossir. 2) Problème de scalabilité 3) Problème de lisibilité 4) Difficillement maintenables

Option 2 : Avec les méthodes sur les listes. (Filter, map, reduce etc)	
Avantages : 1) Performant lorsque la bdd est conséquente 2) Scalabilité 3) Fonctions faciles à comprendre et donc maintenables	Inconvénients : 1) Lenteur d'éxécution quand la base de donnée est petite 2) Pour certains : problème dans compréhension de certains fonctions avancées de ES6

Algorithme 1 :



En console :

1 recherche (« cho ») sans tag donne une vitesse d'éxécution de

```
simpleResearch/ALGO1: 0.25 ms                                    index.js:2343
```

1recherche (« cho ») avec un tag (ingrédient : « beurre ») donne une vitsse d'éxécution de :

```
simpleResearch/ALGO1: 0.255859375 ms                             index.js:2343
```

Algorithme 2 :

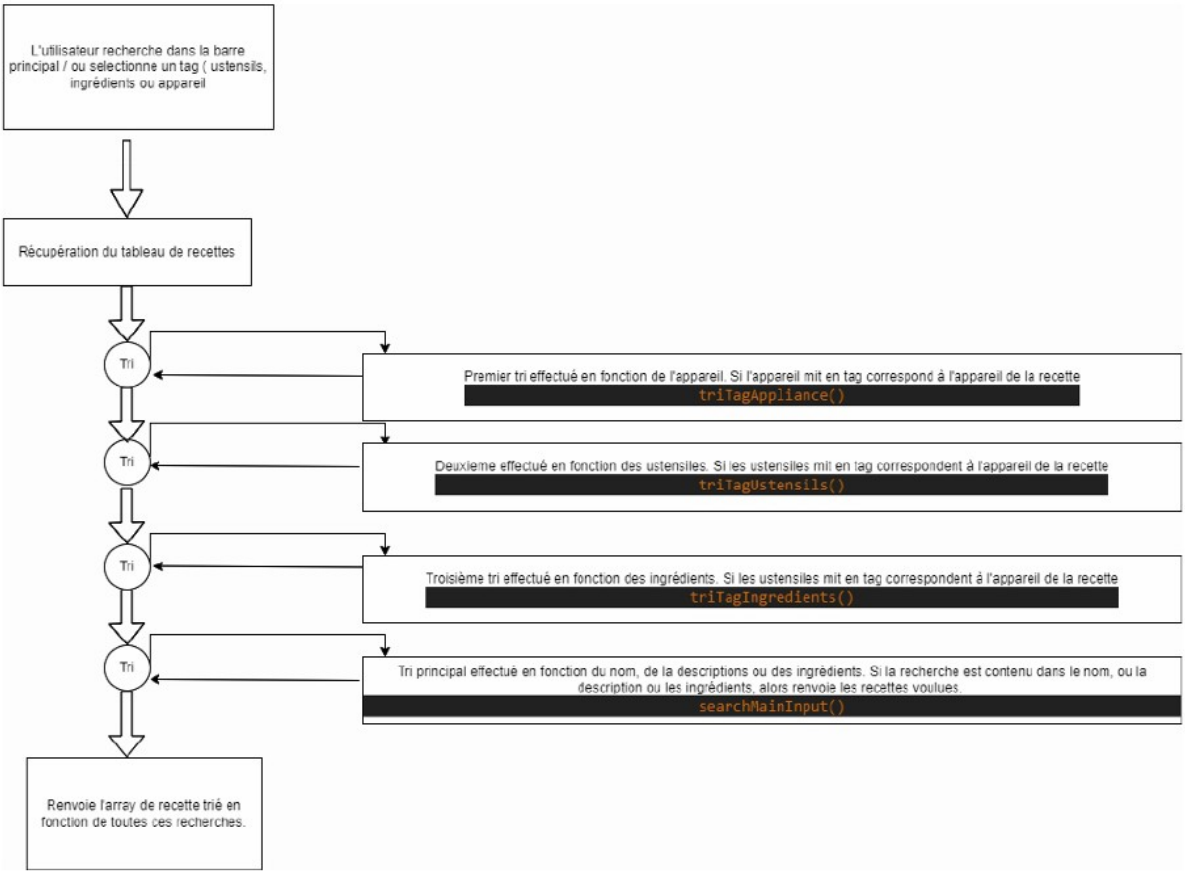


Diagramme similaire au premier algorithme .

En console :

1 recherche (« cho ») sans tag donne une vitesse d'éxécution de

```
triTagAll/ALGO2: 0.578125 ms                                    index.js:2309
```

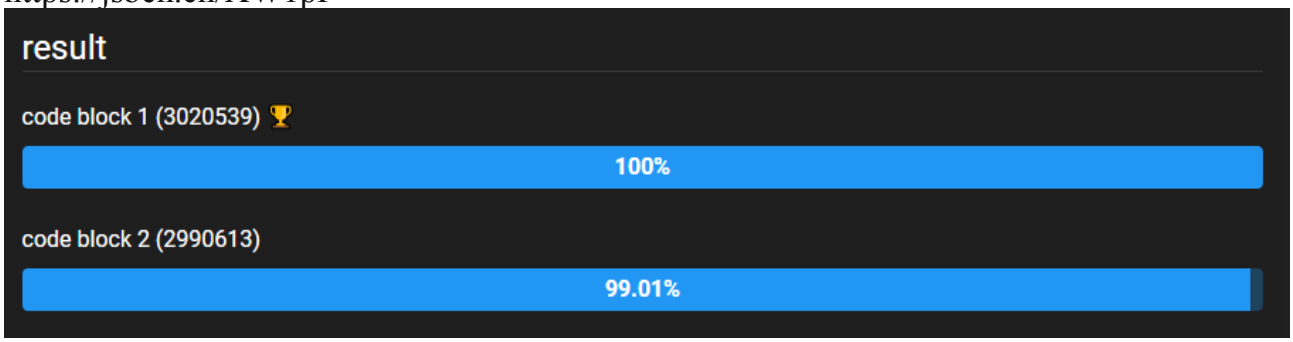
1recherche (« cho ») avec un tag (ingrédient : « beurre ») donne une vitsse d'éxécution de :

```
triTagAll/ALGO2: 0.239990234375 ms                              index.js:2309
```

Solution retenue :

Pour cette application, j'ai choisie de retenir la solution avec l'algorithme numéro 2.

En effet, malgré les indications de JSBENCH.CH
https://jsben.ch/XW1pF



L'algorithme 1 semble être plus rapide de très peu, mais est cependant nettement moins maintenable que le deuxième algorithme.

De plus, la base de donnée ne contenant que 50 recette exactement, il semble normal que les anciennes méthodes de type for/while soient plus performantes, cependant si la base de donnée venait à grossir, l'algoritme numéro 2 serait plus performant.