

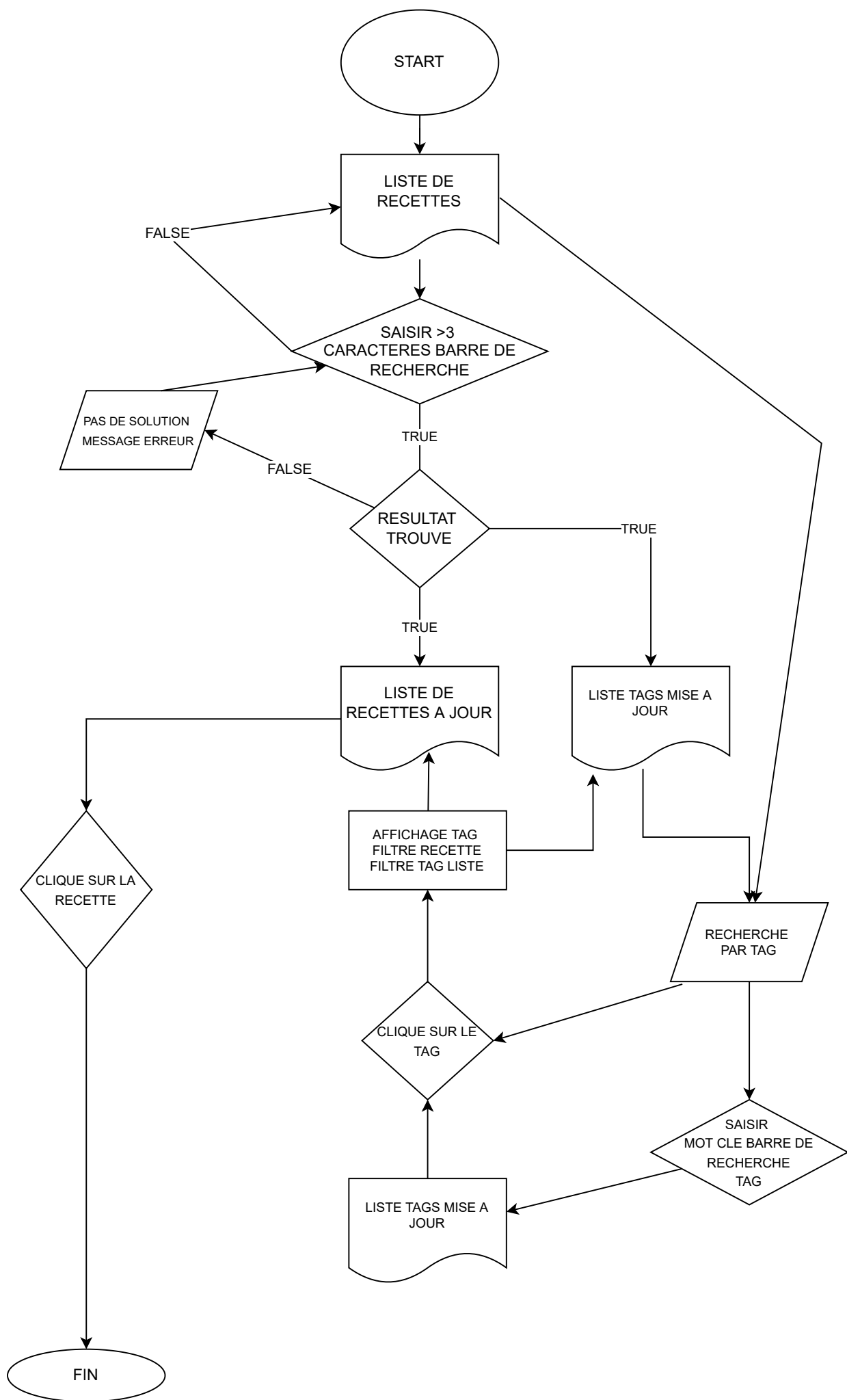
Fiche d'investigation de fonctionnalité

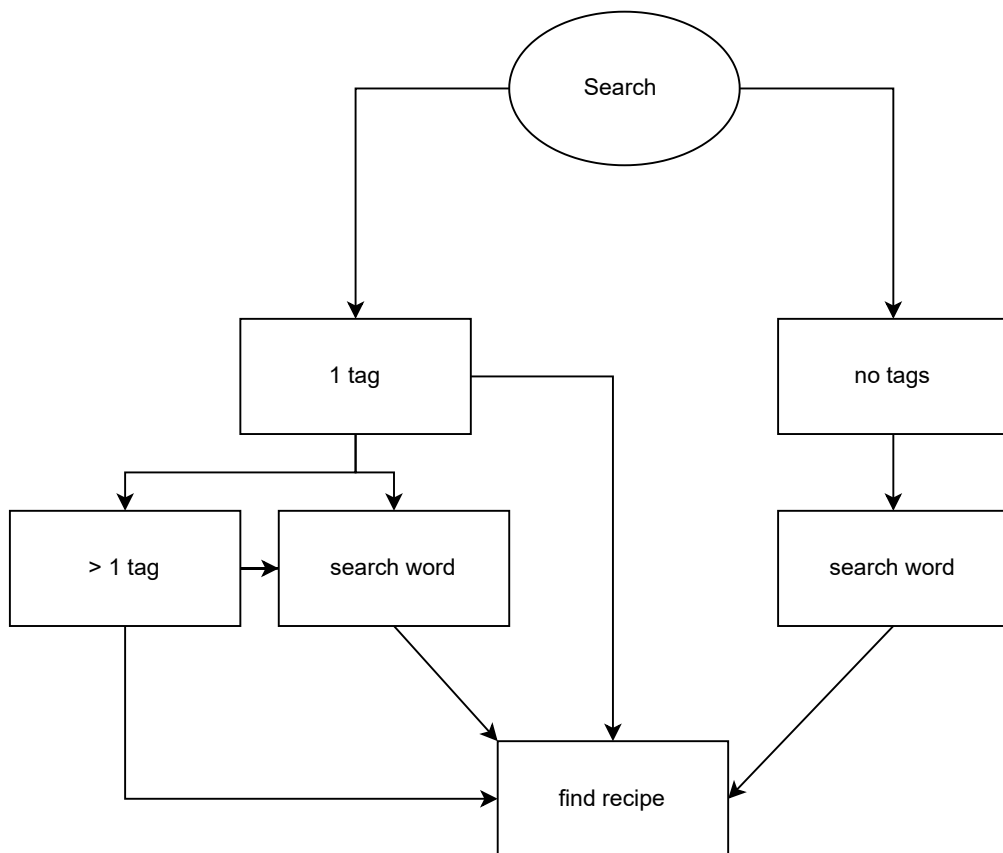
Fonctionnalité : Algorithme de recherche	Fonctionnalité #1
Problématique : Accéder rapidement à une recette correspondant à un besoin de l'utilisateur dans les recettes déjà reçues	

Option 1 : Algorithme utilisant la méthode de l'objet Array "filter"	
Avantages Lisibilité du code Moins de risque d'erreurs accidentelles Implémentation de nouveau filtre rapide	Inconvénients Compatibilité avec les anciens navigateurs

Option 2 : Algorithme utilisant "for" comme boucle native	
Avantages Compatibilité avec les anciens navigateurs	Inconvénients Lisibilité du code Risque d'erreurs accidentelles dues à la complexité du code Maintenabilité moins rapide

Solution retenue :	Suite aux tests réalisés entre les deux méthodes sur JSBench.ch, l'option 1 a été retenue. Elle montre une rapidité accrue de +13% par rapport à l'option 2. 57144 op/s contre 49324 op/s De plus elle est également plus facilement maintenable et son implémentation plus aisée.
---------------------------	---

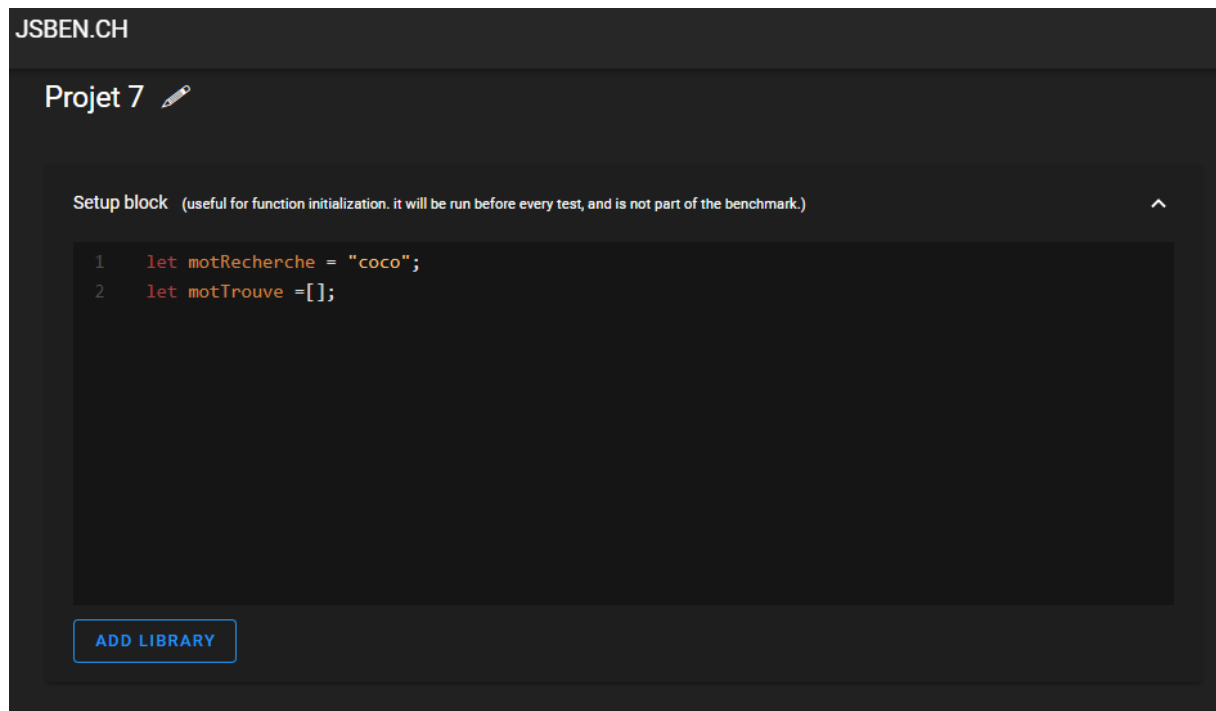




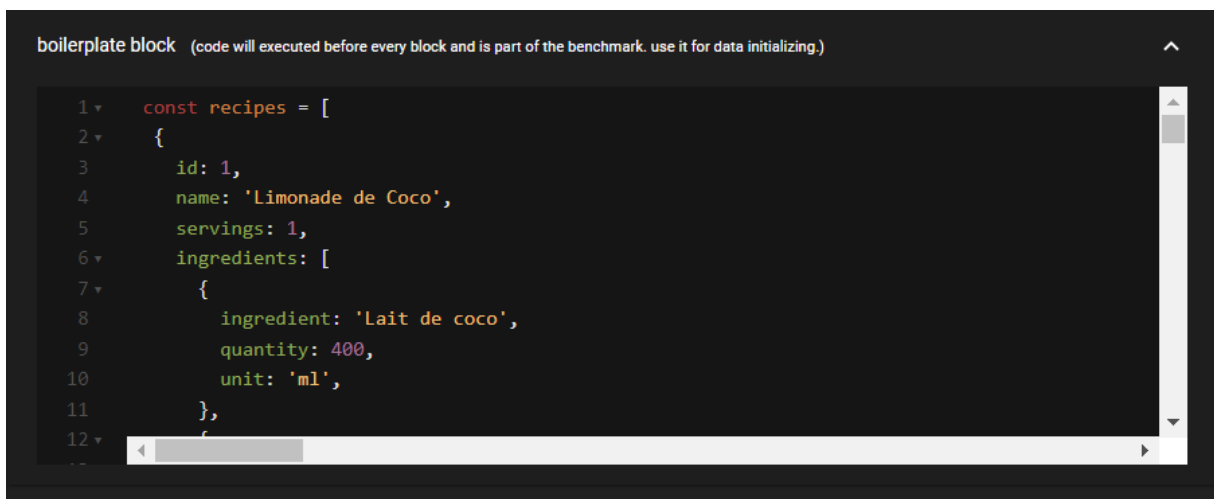
Pour le test de performance, j'utilise JSBEN.CH.

Je crée une variable `motRecherche` = « coco »,

Et une autre variable `motTrouve` = array




Je renseigne la constante « recipes » contenant les 50 recettes.



Je code ma recherche principale.


La première avec la méthode « filter »

La deuxième avec une boucle native « for »

code avec Filter 

```
1  motTrouve = recipes.filter(  
2    (el) =>  
3      el.name.toLowerCase().includes(motRecherche) ||  
4      el.description.toLowerCase().includes(motRecherche) ||  
5      el.ingredients.find((unIngredient) =>  
6        unIngredient.ingredient.toLowerCase().includes(motRecherche)  
7      )  
8    );  
9  
10 return motTrouve;
```



code avec boucle For 

```
1  
2  for (let i = 0; i < recipes.length; i++) {  
3    if (recipes[i].name.toLowerCase().includes(motRecherche) ||  
4        recipes[i].description.toLowerCase().includes(motRecherche) ||  
5        recipes[i].ingredients.find((unIngredient) =>  
6          unIngredient.ingredient.toLowerCase().includes(motRecherche)  
7        ))  
8      {motTrouve.push(recipes[i]);  
9    }  
10 return motTrouve;  
11
```



result

code avec Filter (57144) 🏆

100%

code avec boucle For (49324)

86.32%