

Fiche d'investigation de fonctionnalité

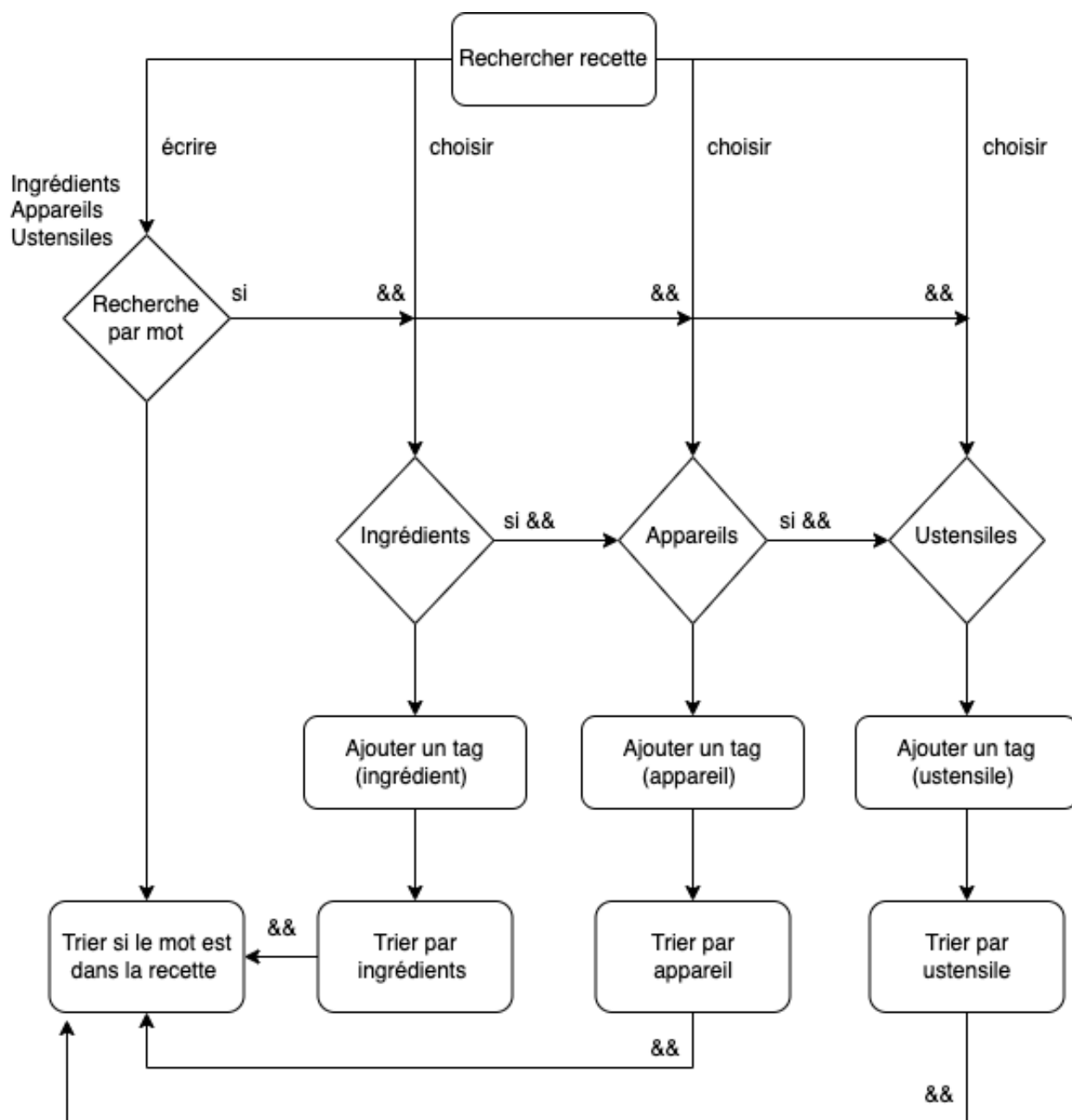
Fonctionnalité : Développez un algorithme de recherche	Fonctionnalité #1
Problématique : Afin de pouvoir faire une recherche plus affinée, nous avons réalisé un algorithme de recherche basé sur le tri des mots des recettes	

Option 1 : Algorithme 1 L'algorithme 1 était la première tentative. Et nous avons décidé de faire une version utilisant les boucles natives (while, for...). L'avantage est qu'il s'agit d'un algorithme plus intuitif pour coder, puisque ce sont des boucles plus simples. L'inconvénient est que le code devient plus long et moins performant, selon jsben.ch.	
Avantages <ul style="list-style-type: none">⊕ Algorithme plus intuitif pour coder⊕ Boucles plus simples dans l'algorithme	Inconvénients <ul style="list-style-type: none">⊖ Le code devient plus long⊖ Le code devient moins performant (selon jsben.ch)
Pourcentage de performance du code, selon jsben.ch : 90.26 %	

Option 2 : Algorithme 2 L'algorithme 2 était la deuxième tentative, nous avons donc essayé de faire une version en programmation fonctionnelle avec les méthodes de l'objet array (foreach, filter, map, reduce). L'avantage est qu'il s'agit d'un code plus propre, avec le moins des boucles possibles et est un code plus performant. L'inconvénient est que c'est un code qui prend plus de temps à écrire, car c'est un peu plus compliqué à coder.	
Avantages <ul style="list-style-type: none">⊕ Un code plus propre⊕ Un code avec le moins des boucles possibles⊕ Un code beaucoup plus performant que l'algorithme 1 selon le site jsben.ch	Inconvénients <ul style="list-style-type: none">⊖ C'est un code qui prend plus de temps à écrire, car c'est un peu plus compliqué à coder.
Pourcentage de performance du code, selon jsben.ch : 100 %	

Solution retenue : Nous avons décidé de choisir l'algorithme 2, en raison de la différence de près de 10 % dans les performances du code.

Annexes



JSBEN.CH

BENCHMARKBROWSEDONATE

no title (put title and/or keywords here, which describes your test)

Setup block

(useful for function initialization. it will be run before every test, and is not part of the benchmark.)

boilerplate block

(code will be executed before every block and is part of the benchmark. use it for data initializing.)

code block 1

```
307
308
309 let recipesHasKeyword = true;
310 if (search.length > 3) {
311   if (!allRecipes[z].name.toLowerCase().includes(search.toL
312   allRecipes[z].description.toLowerCase().includes(search
313   allRecipes[z].ingredients.some(ingredient => ingredient
314     recipesHasKeyword = false;
315   }
316 }
317
318
```

result


code block 2 (4232)

100%

code block 1 (3820)

90.26%

bitbeam



Double
After 1 Week

If you like to donate (Thank you!)