Rapport de Comparaison de Performance des Algorithmes de Recherche de Recettes :

Objectif : Comparer les performances de deux algorithmes de recherche de recettes pour déterminer lequel est le plus efficace.

Méthodologie : Les deux algorithmes ont été testés en utilisant JSBench avec un ensemble de données de recettes. Les résultats des tests ont été mesurés en termes de pourcentage de performance, où l'algorithme 1 a été utilisé comme base de référence (100%).

Problématique : Comment choisir l'algorithme de recherche de recettes le plus efficace qui balance la performance (vitesse d'exécution) et la maintenabilité du code (lisibilité et facilité d'évolution) dans une application web ?

Algorithmes Testés

1. **Algorithme 1**
    - Parcours chaque recette avec plusieurs boucles for.
    - Vérifie si le terme de recherche est présent dans le nom, la description ou les ingrédients.
    - Vérifie chaque mot du terme de recherche dans les ingrédients.
    - Stocke les recettes correspondantes dans un tableau.
2. **Algorithme 2**
    - Utilise Array.prototype.filter pour sélectionner les recettes correspondantes.
    - Vérifie si le terme de recherche est présent dans le nom, la description ou les ingrédients.
    - Utilise Array.prototype.some pour vérifier la présence du terme de recherche dans les ingrédients.
    - Utilise Array.prototype.every pour vérifier chaque mot du terme de recherche dans les ingrédients.

Résultats des Tests

Algorithme	Performance (%)
Algorithme 1	100%
Algorithme 2	98.63%

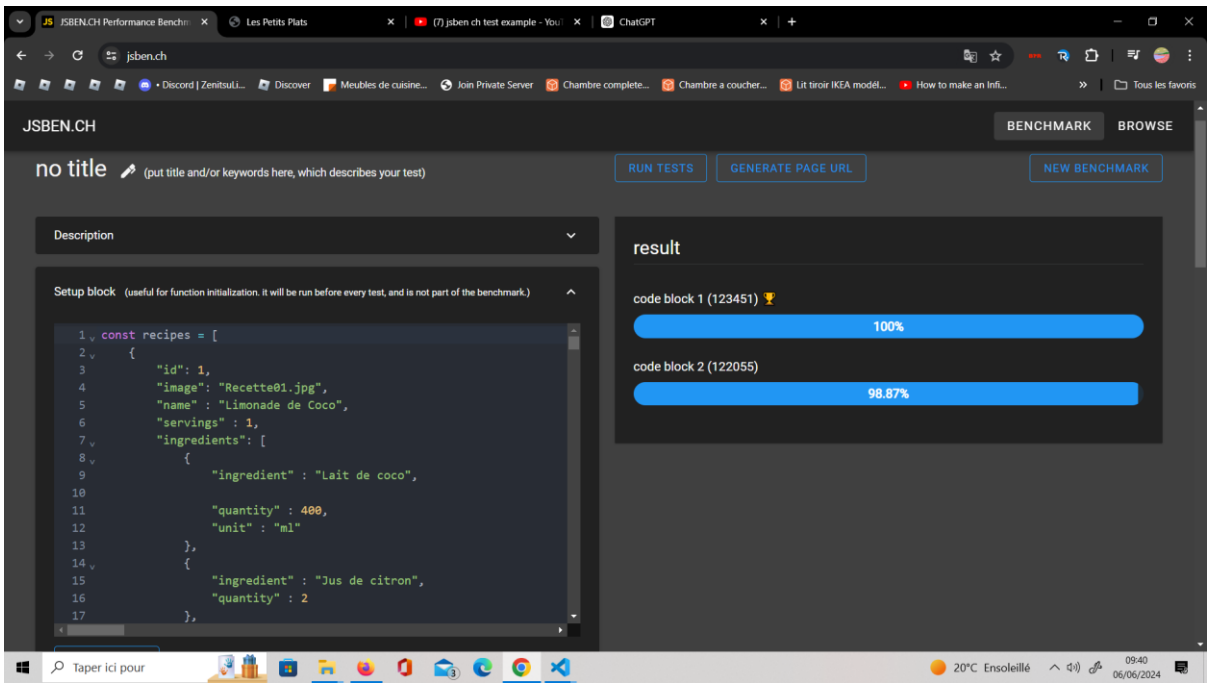
Analyse des Résultats

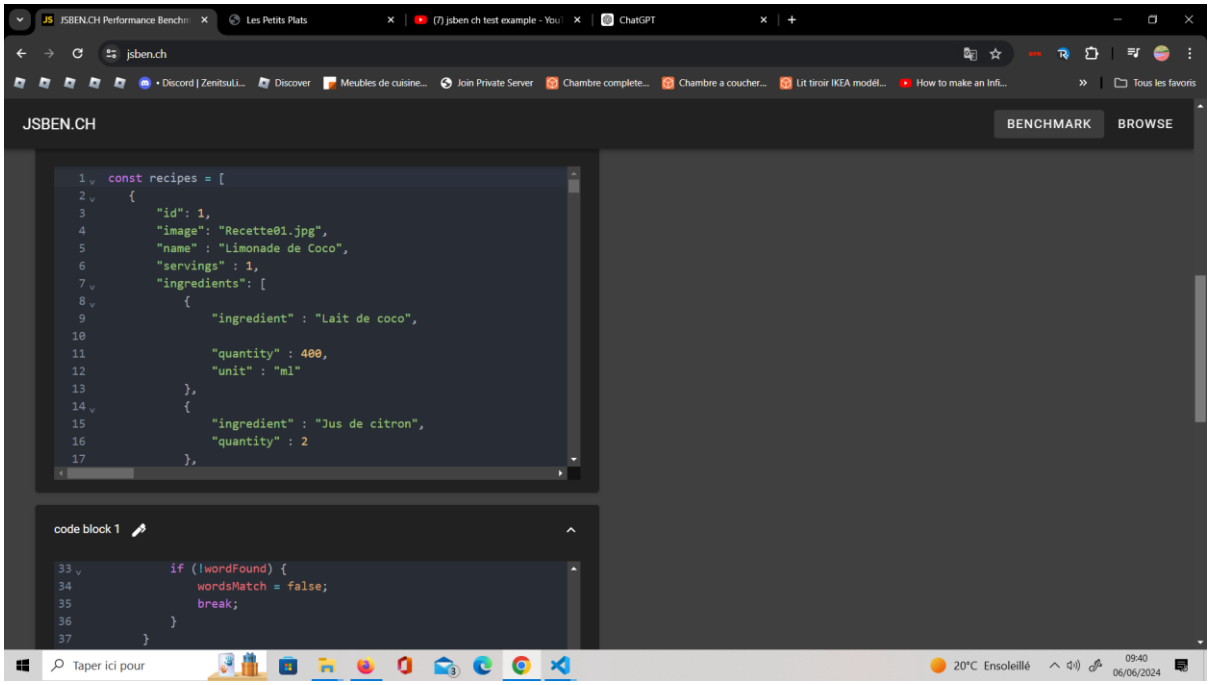
- **Algorithme 1** a été utilisé comme base de référence avec une performance de 100%. Cet algorithme, bien que plus performant, utilise des boucles imbriquées qui peuvent rendre le code plus complexe et moins lisible.
- **Algorithme 2** a obtenu une performance de **98.63%** par rapport à l'algorithme 1. Bien que légèrement moins performant, cet algorithme utilise des méthodes modernes (filter, some, every) qui rendent le code plus concis et plus facile à comprendre.

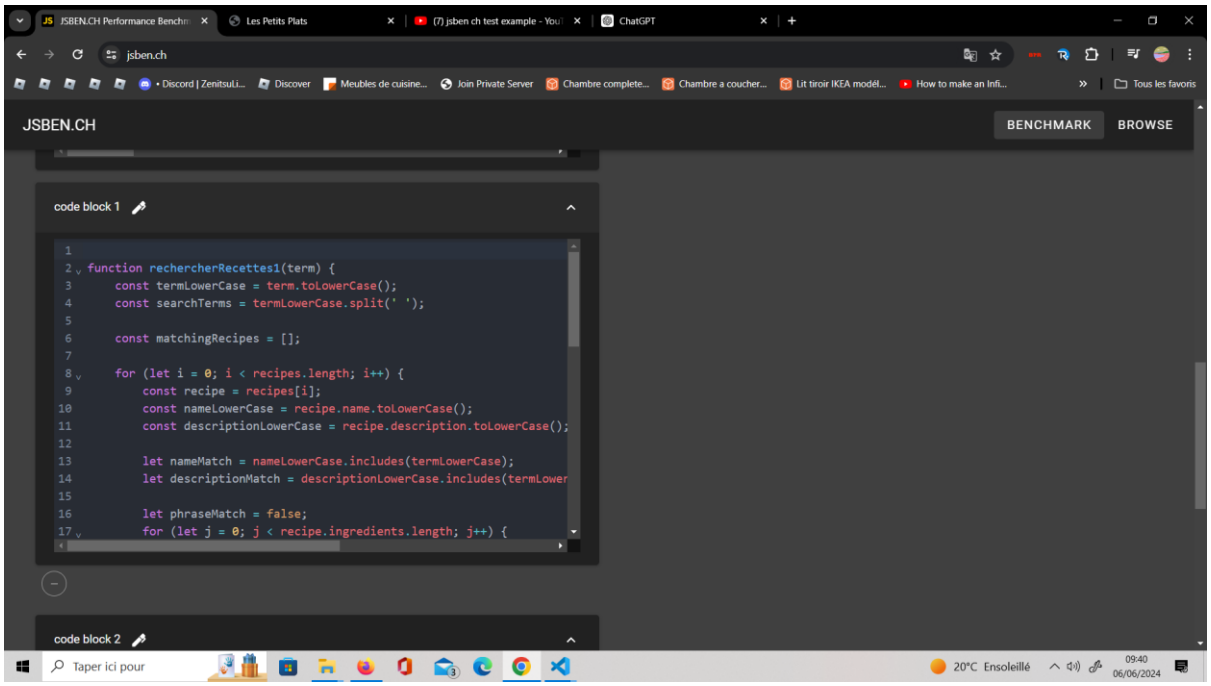
Conclusion

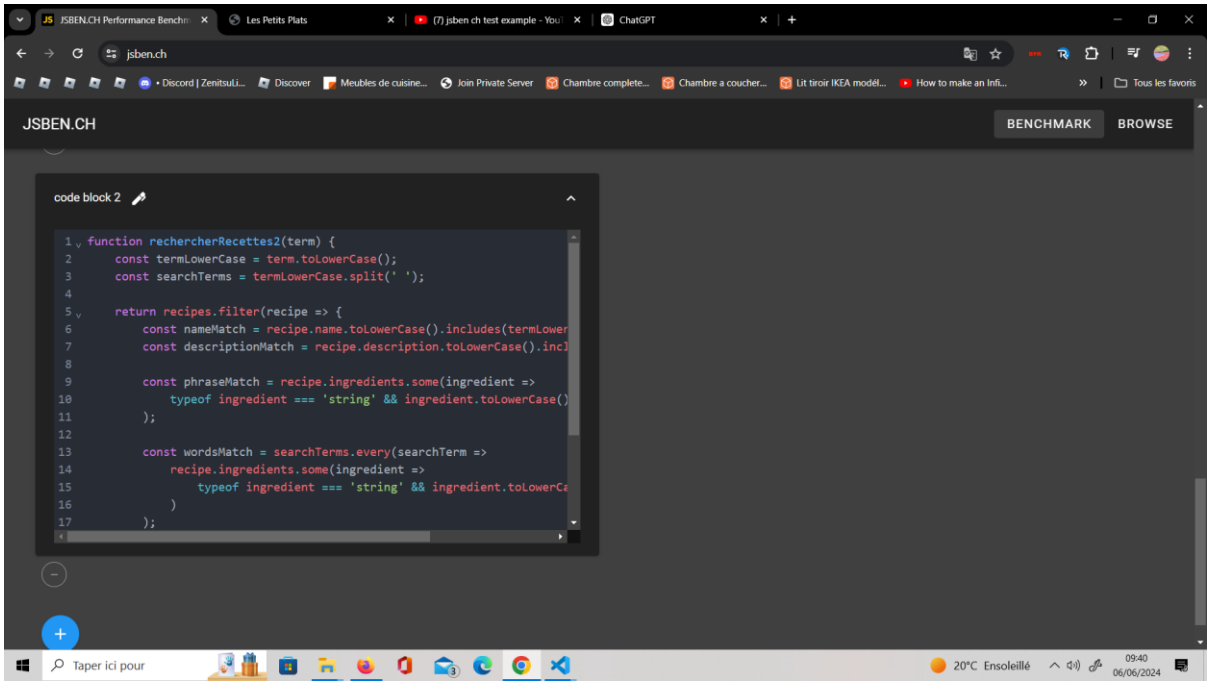
- Les deux algorithmes sont très performants pour la recherche de recettes, avec une différence de performance négligeable (1.37%).
- **Recommandation** : Bien que l'algorithme 1 soit légèrement plus performant, l'algorithme 2 est recommandé pour une utilisation pratique en raison de sa modernité et de sa lisibilité. L'utilisation de méthodes modernes de JavaScript rend le code plus maintenable et adaptable aux évolutions futures.

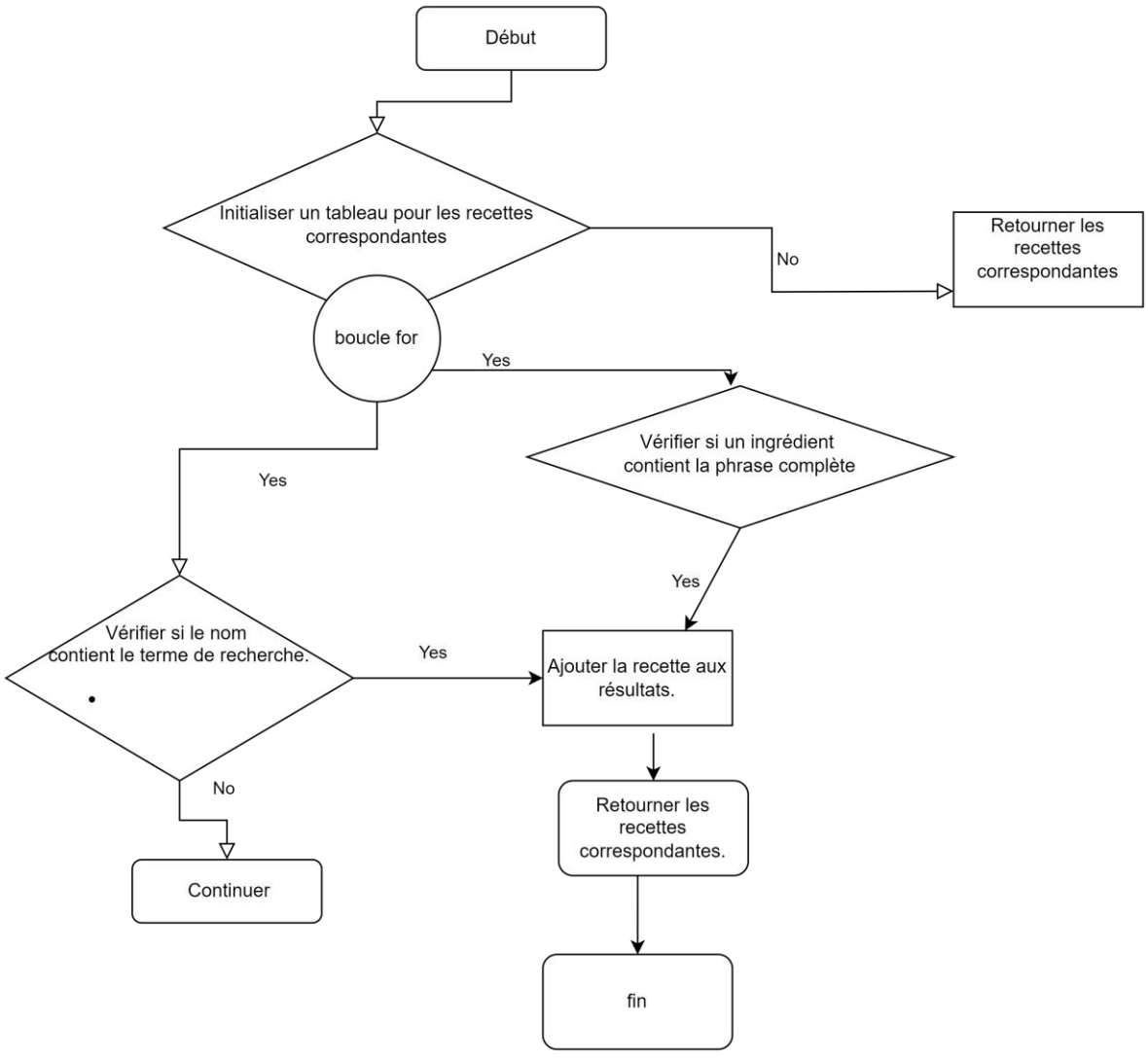
En résumé, pour une balance optimale entre performance et maintenabilité du code, l'algorithme 2 est préférable, malgré une très légère baisse de performance.

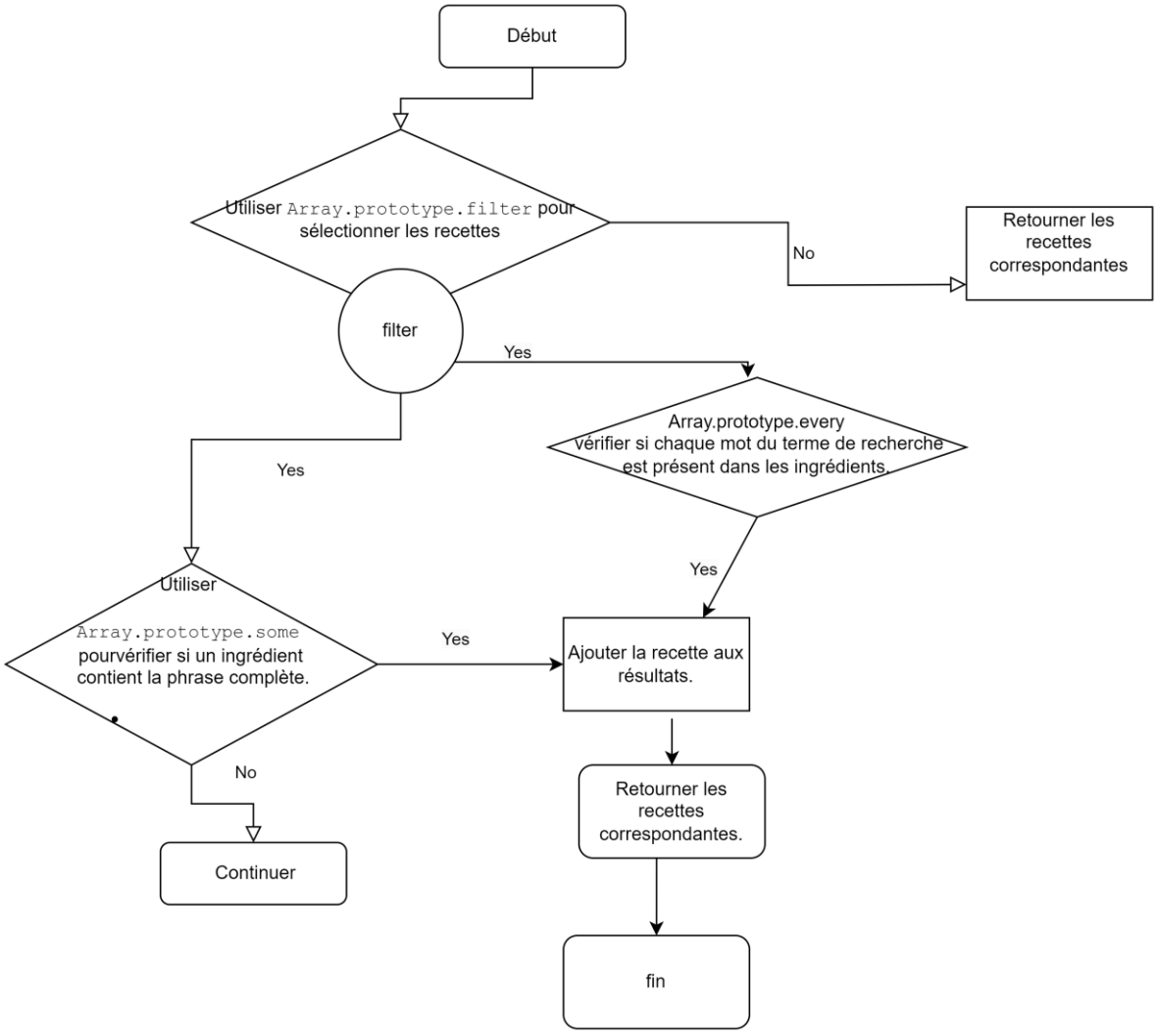



JSBEN.CH Performance Benchm
Les Petits Plats
(7) jsben ch test example - You
ChatGPT
jsben.ch
Discord | ZenitsuLi...
Discover
Meubles de cuisine...
Join Private Server
Chambre complete...
Chambre a coucher...
Lit tiroir IKEA modél...
How to make an Infi...
Tous les favoris
JSBEN.CH
BENCHMARK
BROWSE
const recipes = [
{
"id": 1,
"image": "Recette01.jpg",
"name" : "Limonade de Coco",
"servings" : 1,
"ingredients": [
{
"ingredient" : "Lait de coco",
"quantity" : 400,
"unit" : "ml"
},
{
"ingredient" : "Jus de citron",
"quantity" : 2
},
code block 1
if (!wordFound) {
wordsMatch = false;
break;
}
}
Taper ici pour
20°C Ensoleillé
09:40
06/06/2024


JSBEN.CH Performance Benchm
Les Petits Plats
(7) jsben ch test example - You
ChatGPT
jsben.ch
Discord | ZenitsuLi...
Discover
Meubles de cuisine...
Join Private Server
Chambre complete...
Chambre a coucher...
Lit tiroir IKEA modél...
How to make an Infi...
Tous les favoris
JSBEN.CH
BENCHMARK
BROWSE
code block 1
function rechercherRecettes1(term) {
const termLowerCase = term.toLowerCase();
const searchTerms = termLowerCase.split(' ');
const matchingRecipes = [];
for (let i = 0; i < recipes.length; i++) {
const recipe = recipes[i];
const nameLowerCase = recipe.name.toLowerCase();
const descriptionLowerCase = recipe.description.toLowerCase();
let nameMatch = nameLowerCase.includes(termLowerCase);
let descriptionMatch = descriptionLowerCase.includes(termLower
let phraseMatch = false;
for (let j = 0; j < recipe.ingredients.length; j++) {
code block 2
Taper ici pour
20°C Ensoleillé
09:40
06/06/2024


JSBEN.CH
BENCHMARK
BROWSE
code block 2
function rechercherRecettes2(term) {
const termLowerCase = term.toLowerCase();
const searchTerms = termLowerCase.split(' ');
return recipes.filter(recipe => {
const nameMatch = recipe.name.toLowerCase().includes(termLower
const descriptionMatch = recipe.description.toLowerCase().incl
const phraseMatch = recipe.ingredients.some(ingredient =>
typeof ingredient === 'string' && ingredient.toLowerCase()
);
const wordsMatch = searchTerms.every(searchTerm =>
recipe.ingredients.some(ingredient =>
typeof ingredient === 'string' && ingredient.toLowerCa
)
);


Début
Initialiser un tableau pour les recettes correspondantes
No
Retourner les recettes correspondantes
boucle for
Yes
Vérifier si un ingrédient contient la phrase complète
Yes
Yes
Vérifier si le nom contient le terme de recherche.
Yes
Ajouter la recette aux résultats.
No
Retourner les recettes correspondantes.
Continuer
fin


Début
Utiliser Array.prototype.filter pour sélectionner les recettes
No
Retourner les recettes correspondantes
filter
Yes
Array.prototype.every
Vérifier si chaque mot du terme de recherche est présent dans les ingrédients.
Yes
Yes
Utiliser
Array.prototype.some
pourvérifier si un ingrédient contient la phrase complète.
Yes
Ajouter la recette aux résultats.
No
Retourner les recettes correspondantes.
Continuer
fin