Fiche d’investigation de fonctionnalité

Fonctionnalité : Search recettes	**Fonctionnalité #2**
Problématique : Accéder rapidement à une recette correspondant à un besoin de l’utilisateur dans les recettes déjà reçues	

Option 1 : « ES6 » native
Recherche à l'aide des nouvelles méthodes natives JS pour les tableaux et les chaînes de caractères.

Avantages	**Inconvénients**
• Une approche impérative • Code plus moderne et plus orienté vers l'avenir	• Risque de ne pas être compatible avec les anciens navigateurs

Nombre de paramètres d'entrée : 2
recipes - un tableau d'objets DataModelRecipe représentant chaque recette dans la base de données
searchInput - la saisie de l'utilisateur dans la barre de recherche principale sous la forme d'une chaîne de caractères

Résultat de la fonction :
Un tableau contenant les valeurs booléennes (True/False) représentant chacune si une recette correspond ou non à l'expression recherchée.

Option 2 : Boucles « For » classiques et « Regex »
Utiliser la boucle For classique pour tester l'expression Regex sur les titres, les descriptions et les listes d'ingrédients de chaque recette.

Avantages	**Inconvénients**
• Une approche déclarative • Une solution plus classique et bien connue	• Code moins lisible et plus difficile à analyser avec une boucle imbriquée

Nombre de paramètres d'entrée : 2
recipes - un tableau d'objets DataModelRecipe représentant chaque recette dans la base de données
searchInput - la saisie de l'utilisateur dans la barre de recherche principale sous la forme d'une chaîne de caractères

Résultat de la fonction :
Un tableau contenant les valeurs booléennes (True/False) représentant chacune si une recette correspond ou non à l'expression recherchée.

Solution retenue :
Après les tests JSBen, je recommande d'utiliser le premier algorithme car il est plus rapide.

Annexes

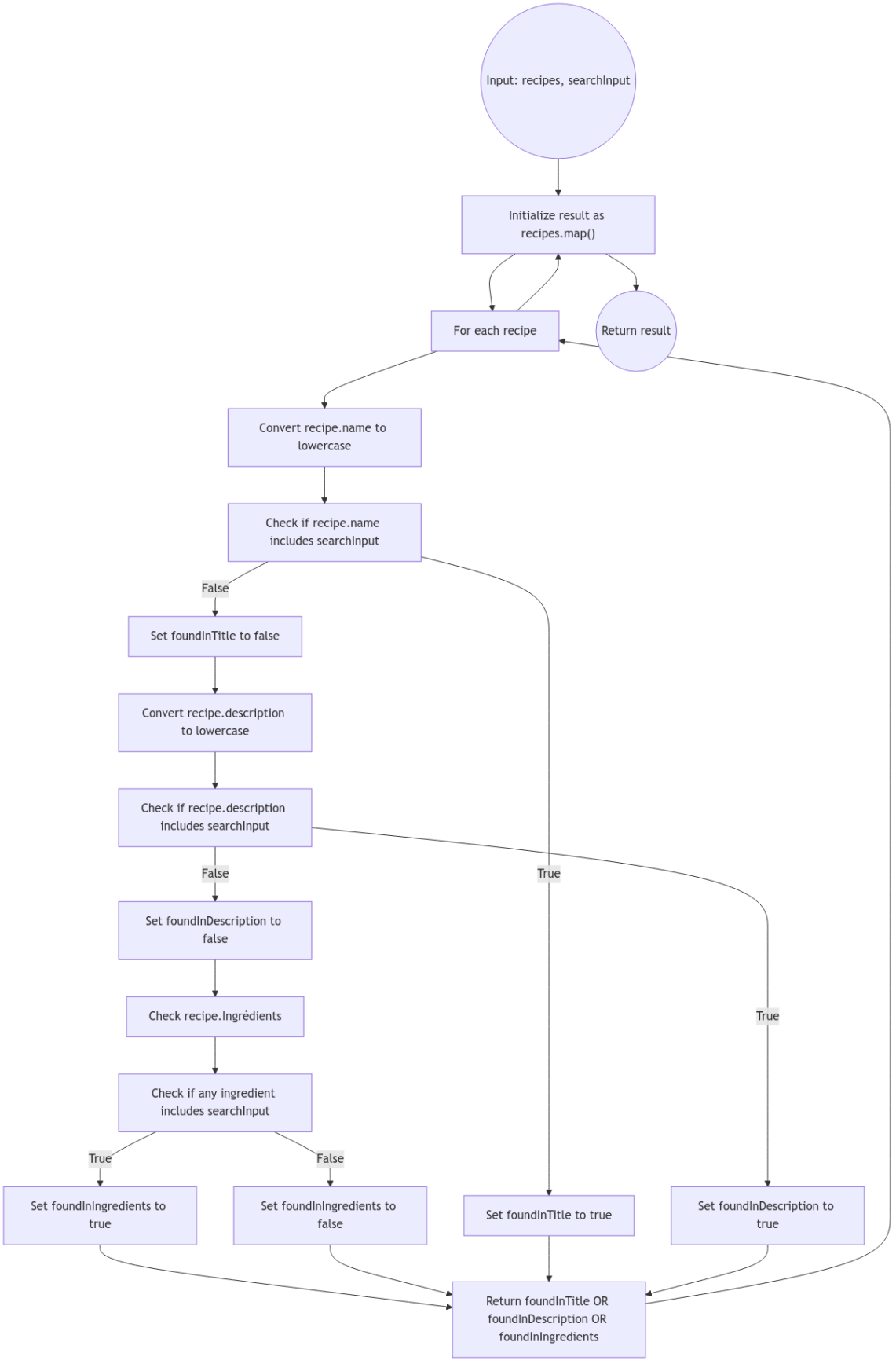


Figure 1 - Diagramme « ES6 » native

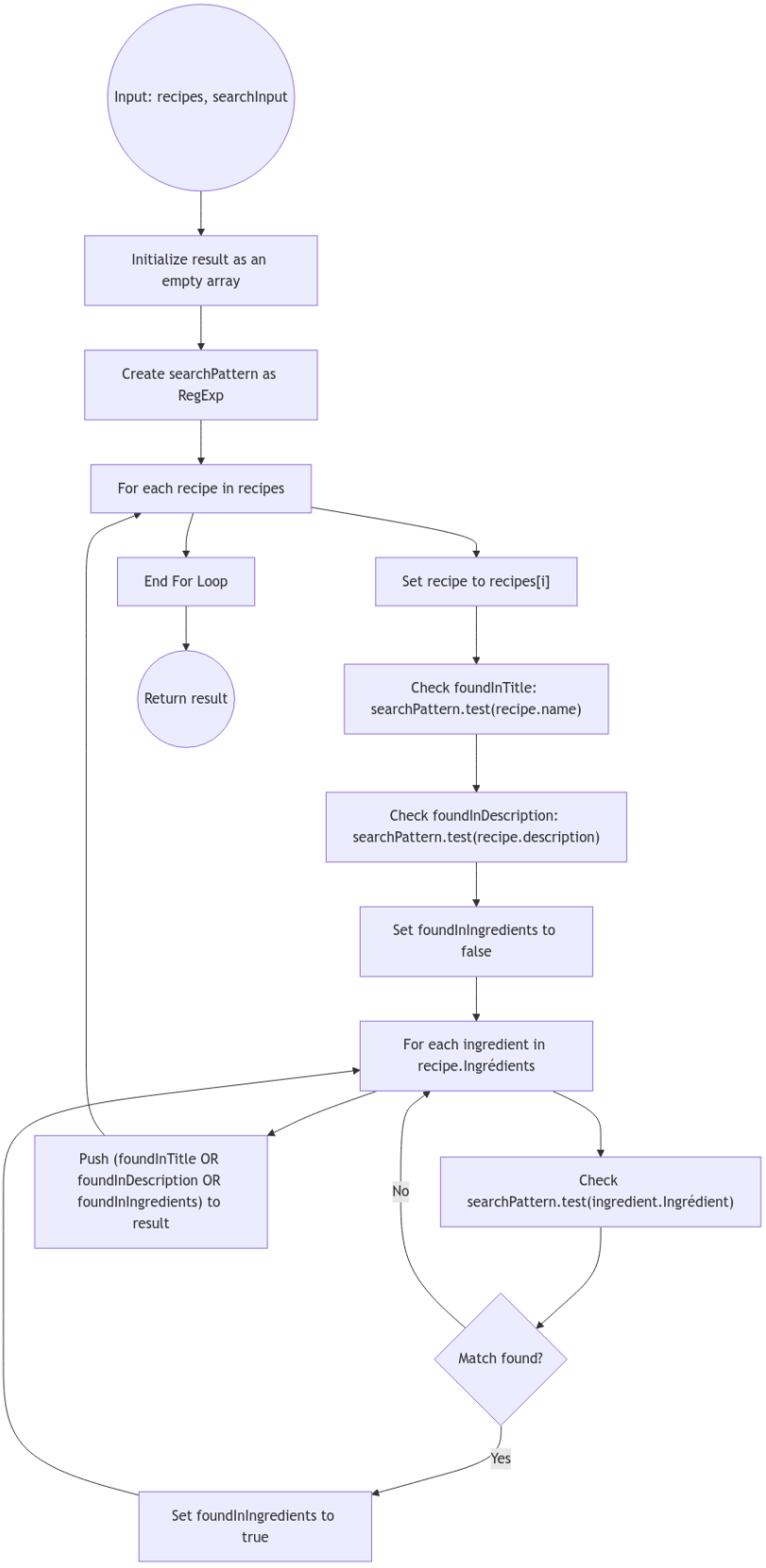
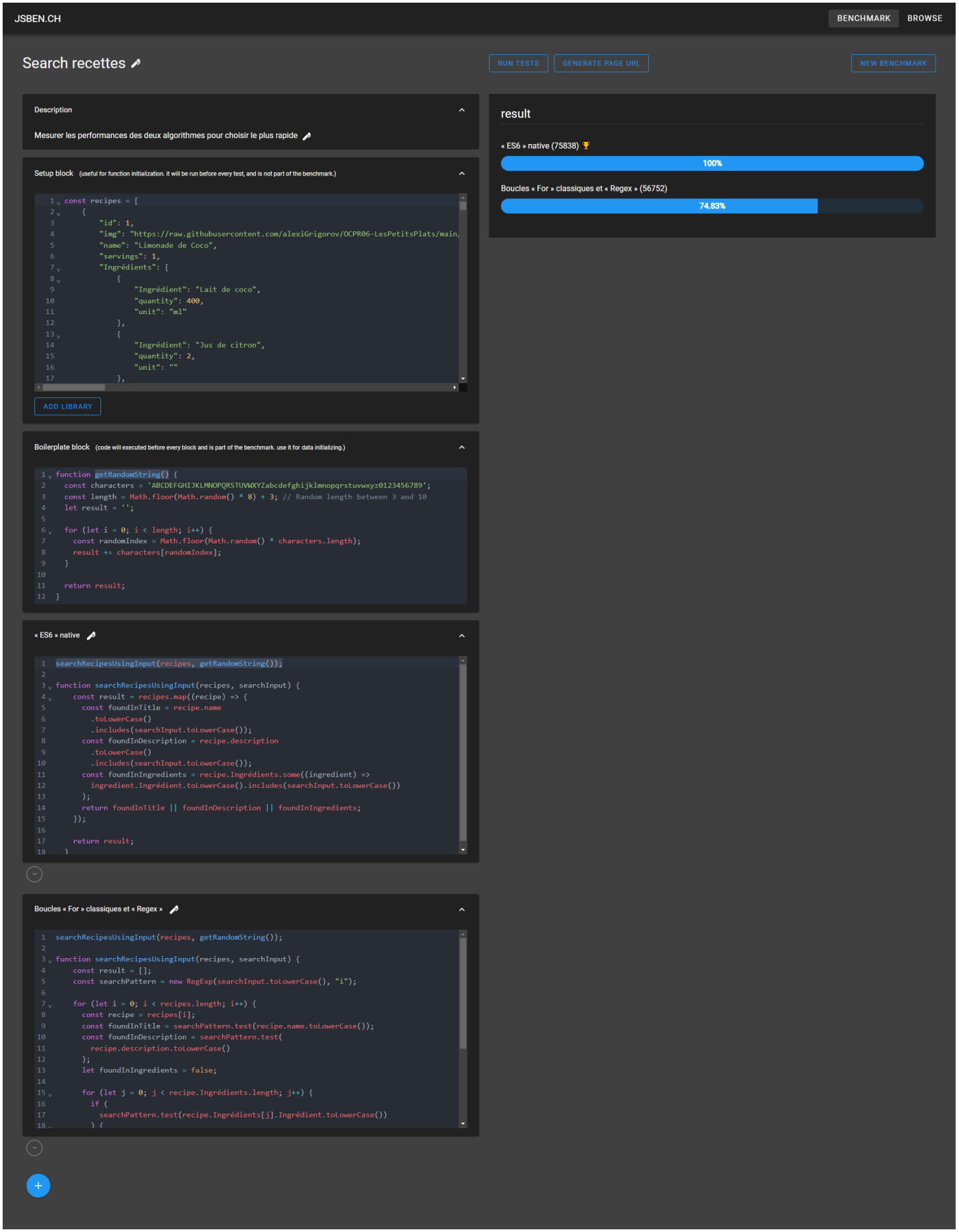


Figure 2 : Boucles « For » classiques et « Regex »


JSBEN.CH
BENCHMARK
BROWSE
Search recettes
RUN TESTS
GENERATE PAGE URL
NEW BENCHMARK
Description
Mesurer les performances des deux algorithmes pour choisir le plus rapide
Setup block (useful for function initialization. it will be run before every test, and is not part of the benchmark.)
const recipes = [
{
"id": 1,
"img": "https://raw.githubusercontent.com/alexiGrigorov/OCPR06-LesPetitsPlats/main,
"name": "Limonade de Coco",
"servings": 1,
"Ingrédients": [
{
"Ingrédient": "Lait de coco",
"quantity": 400,
"unit": "ml"
},
{
"Ingrédient": "Jus de citron",
"quantity": 2,
"unit": ""
},
ADD LIBRARY
Boilerplate block (code will executed before every block and is part of the benchmark. use it for data initializing.)
function getRandomString() {
const characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
const length = Math.floor(Math.random() * 8) + 3; // Random length between 3 and 10
let result = '';
for (let i = 0; i < length; i++) {
const randomIndex = Math.floor(Math.random() * characters.length);
result += characters[randomIndex];
}
return result;
}
« ES6 » native
searchRecipesUsingInput(recipes, getRandomString());
function searchRecipesUsingInput(recipes, searchInput) {
const result = recipes.map((recipe) => {
const foundInTitle = recipe.name
.toLowerCase()
.includes(searchInput.toLowerCase());
const foundInDescription = recipe.description
.toLowerCase()
.includes(searchInput.toLowerCase());
const foundInIngredients = recipe.Ingrédients.some((ingredient) =>
ingredient.Ingrédient.toLowerCase().includes(searchInput.toLowerCase())
);
return foundInTitle || foundInDescription || foundInIngredients;
});
return result;
}
Boucles « For » classiques et « Regex »
searchRecipesUsingInput(recipes, getRandomString());
function searchRecipesUsingInput(recipes, searchInput) {
const result = [];
const searchPattern = new RegExp(searchInput.toLowerCase(), "i");
for (let i = 0; i < recipes.length; i++) {
const recipe = recipes[i];
const foundInTitle = searchPattern.test(recipe.name.toLowerCase());
const foundInDescription = searchPattern.test(
recipe.description.toLowerCase()
);
let foundInIngredients = false;
for (let j = 0; j < recipe.Ingrédients.length; j++) {
if (
searchPattern.test(recipe.Ingrédients[j].Ingrédient.toLowerCase())
) {
result
« ES6 » native (75838)
100%
Boucles « For » classiques et « Regex » (56752)
74.83%