

Fiche d'investigation de fonctionnalité

Fonctionnalité : Barre principale de recherche

Fonctionnalité #1

Problématique : Afin que les utilisateurs puissent trouver une recette facilement, nous cherchons à accéder rapidement à une recette correspondant à un besoin de l'utilisateur dans les recettes déjà reçues.

Option 1 : Programmation native avec la méthode For (Annexe)

On emploie ici la méthode « for » qui crée une boucle de recherche dans le tableau des recettes et cherche s'il existe une correspondance entre la saisie du nom de la recette, le matériel ou un des ingrédients de la recette.

Si oui, la recette en question est ajoutée à un nouveau tableau qui servira à l'affichage des recettes trouvées.

De ce tableau, les différentes listes sont mises à jour également.

Avantages

- + code plus lisible
- + syntaxe plus facile à comprendre côté frontend et backend

Inconvénients

- + le code est lent
- + le code n'est pas du tout abstrait

Nombre de champs minimum dans la barre de recherche : 3 lettres

Option 2 : Programmation fonctionnelle avec la méthode Filter (Annexe)

On emploie ici la méthode « filter » qui filtre les recettes suivant la saisie effectuée et les correspondances trouvées dans le nom de la recette, le matériel ou les ingrédients de la recette.

La recette trouvée est ajoutée à un tableau qui servira à l'affichage des recettes.

De ce tableau, les différentes listes sont mises à jour.

Avantages

- + Filtrage rapide
- + système robuste et fiable
- + retourne un nouveau tableau au lieu de le modifier

Inconvénients

- + le code est trop abstrait

Nombre de champs minimum dans la barre de recherche : 3 lettres

Solution retenue :

Test avec [JsBench.me](https://jsbench.me) :

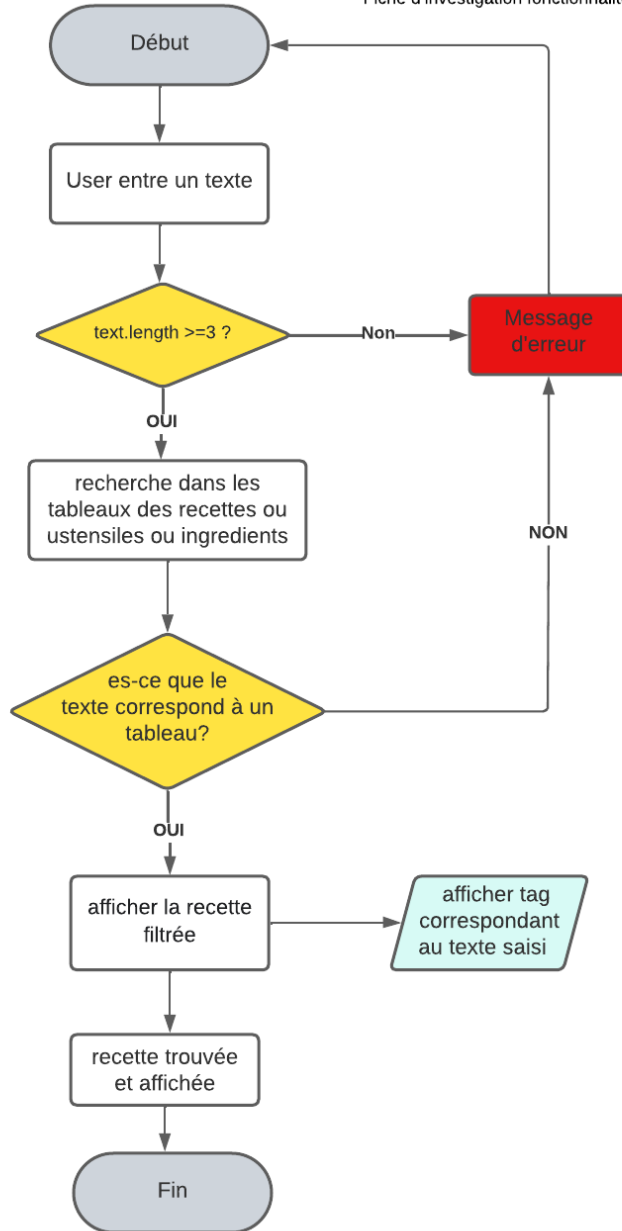
- la programmation fonctionnelle semble être la plus rapide d'après le test Jsbench
- la programmation native avec « for » est plus lente qu'avec un filter.

Test avec console.time() :

Avec for timer: 0.0087890625 ms en moyenne

Avec filter timer: 0.007080078125 ms en moyenne

Nous avons donc retenu l'option : 2



JSBench.Me

Run

Please login/register to save & publish tests

Sponsor

Settings

Sign in

test algorithme souad mouatakide

v1

- by

enter test suite description

Setup HTML

```
<body>
  <header>
    <div class="container-field p-3 pb-2">
      <img logo="logo" />
      <div class="row align-items-center">
        <div class="col text-sm-center">
          
        </div>
      </div>
    </div>
  </body>
```

Setup JavaScript

```
//ingredients
const ingredientsChevron = document.querySelector("#ingredientsChevron");
const appliancesChevron = document.querySelector("#appliancesChevron");
const utensilsChevron = document.querySelector("#utensilsChevron");

//NA
const allIngredients = document.querySelector("#ingredientsList");
const allAppliances = document.querySelector("#appliancesList");
const allUtensils = document.querySelector("#utensilsList");

//cartes de recettes
const recipesContainer = document.querySelector("#output");
```

algo 1 filter

```
function principalRecipeFilter(recipesToFilter) {
  let selectedRecipesSearch = [];
  recipesToFilter.forEach(recipe => {
    if (
      recipe.name
        .toLowerCase()
        .replace(/<\/g, >>
        .includes(principalRecipeSearchValue) ||
      recipe.description
        .toLowerCase()
        .replace(/<\/g, >>
        .includes(principalRecipeSearchValue) ||
    ) {
      selectedRecipesSearch.push(recipe);
    }
  });
  return selectedRecipesSearch;
}
```

finished

84385539.57 ops/s ± 1.3%

Fastest

algo 2 for

```
function principalRecipeFilter(recipesToFilter) {
  let selectedRecipesSearch = [];
  for (let recipe of recipesToFilter) {
    for (let i = 0; i < recipe.ingredients.length; i++) {
      const ingredientName = recipe.ingredients[i].ingredient
        .toLowerCase()
        .replace(/<\/g, >>
      if (ingredientName.includes(principalRecipeSearchValue)) {
        selectedRecipesSearch.push(recipe);
      }
    }
  }
  return selectedRecipesSearch;
}
```

finished

83074743.85 ops/s ± 2.46%

Fastest