Fiche d'Investigation Fonctionnalité

Fonctionnalité : Rechercher des recettes selon des critères de recherche de mots-clés et de tags

Problématique : Pourvoir voir la recette désirée selon mots-clés initiaux, et tags ingrédients, ustensiles et appareils. Nombre de champs minimum à remplir dans la barre de recherche : 3. Pour plus d'informations concernant le déroulement du rendu Front, référez-vous à l'algorigramme.

Option 1 : Prototype Array

Option pour obtenir la combinaison de toutes les recettes avec des prototypes d'array JavaScript. L'avantage c'est d'utiliser les méthodes modernes avec du code concis qui suit les standards de coding JS standardisé.

Avantages :

- Utilisation des méthodes modernes
- Code concis et standardisé

Inconvénients :

- Moins de contrôle sur le code employé

Option 2 : Boucle JS

Option pour obtenir la combinaison de toutes les recettes avec les méthodes Boucles JS. L'avantage c'est d'avoir plus de contrôle sur le code employé pour récupérer les informations désirées.

Avantages :

- Plus de contrôle sur le code

Inconvénients :

- Code potentiellement plus verbeux

Solution retenue :

L'option des boucles natives JS est la plus performante. D'un point de vue vitesse nous choisirons donc l'option des boucles natives. En revanche, l'écart de performances entre l'option Prototype.Array et boucles JS natifs n'ont pas un écart significatif. D'un point de vue maintenabilité et simplicité du code, nous pourrions très bien choisir la méthode Prototype.Array.

Lien vers le benchmark

Benchmark JS - Prototype.Array vs Boucles JS

RUN TESTS | GENERATE PAGE URL | NEW BENCHMARK

Description

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

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

```
const recipes = [
  {
    id: 1,
    image: 'Recette01.webp',
    name: 'Limonade de Coco',
    servings: 1,
    ingredients: [
      {
        ingredient: 'Lait de coco',
        quantity: 400,
        unit: 'ml',
      },
      {
        ingredient: 'Jus de citron',
        quantity: 2,
      },
      {
        ingredient: 'Crème de coco',
```

code block 1

```
    }

    if (searchQueryParamsNormalized.length >= 3) {
      const recipeString = JSON.stringify(recipe, null);
      const recipeNormalized = normalizeString(recipeString);
      return recipeNormalized.includes(searchQueryParamsNormalized)
    }
  };

  const getFilteredRecipes = (queryParams = getQueryParams()) => {
    const filteredRecipes = recipes.filter((recipe) => {
      return (
        getFilteredServices(recipe, queryParams.search)
      );
    });

    return filteredRecipes;
  };
```

code block 2

```
  const getFilteredServices = (recipe, searchParam) => {
    const normalizeString = (str) => {
      return str
        .toLowerCase()
        .normalize('NFD')
        .replace(/[\u0300-\u036f]/g, '')
        .replace(/[^a-zA-Z0-9\s]/g, '');
    };

    const searchQueryParamsNormalized = normalizeString(searchParam
    if (!searchQueryParamsNormalized) {
      return true;
    }

    if (searchQueryParamsNormalized.length >= 3) {
      const recipeString = JSON.stringify(recipe, null);
      const recipeNormalized = normalizeString(recipeString);
      return recipeNormalized.includes(searchQueryParamsNormalized)
```

result

code block 1 (1191649) 🏆

100%

code block 2 (1177742)

98.83%

Algorigramme :