

Fiche d’investigation de fonctionnalité

|  |  |
| --- | --- |
| **Fonctionnalité :**   * Affichage dynamique des recettes. * Barre de recherche principale (min. 3 caractères). * Filtres par tags (ingrédients, appareils, ustensiles). * Suppression des tags → mise à jour de la liste. * Gestion des doublons dans les listes. * Compteur de recettes affichées. | **Fonctionnalité #1** |
| **Problématique :**  Afin de permettre à l’utilisateur de trouver rapidement une recette parmi plus de 1500 propositions, il faut mettre en place une recherche efficace combinée à des filtres (ingrédients, appareils, ustensiles). L’objectif est d’éviter que l’utilisateur se perde dans une liste trop longue et de lui donner un système fluide, rapide et intuitif. | |

|  |  |
| --- | --- |
| **Implémentation 1 :** **Méthodes modernes (forEach, filter, some, Set)**   * **Extraction des données :** forEach pour parcourir les recettes. / Set pour supprimer les doublons. * **Filtrage :** filter pour appliquer la recherche principale. / some pour vérifier la présence d’un ingrédient ou ustensile. * **Affichage :** Construction dynamique du DOM avec forEach. | |
| **Avantages :**   * Code plus concis et lisible. * Méthodes optimisées par le moteur JS. * Idéal pour un projet moderne (clarté, maintenabilité). * Boucle idéale pour parcourir les tableaux qui ne consistent pas d’indexation | **Inconvénients :**   * Difficile pour débuter * Avoir connaissance des méthodes ES6+ |

|  |  |
| --- | --- |
| **Implémentation 2 :** **Méthodes Classique (for)**   * **Extraction des données :** Boucles for imbriquées pour parcourir recettes, ingrédients et ustensiles. / Fonction removeDuplicates() avec deux boucles for pour éliminer les doublons. * **Filtrage :** Recherche principale avec indexOf. / Vérification des tags via boucles imbriquées. * **Affichage :** Construction dynamique du DOM avec for. | |
| **Avantages :**   * Très explicite → chaque étape est visible. * Ne nécessite pas de connaissances avancées (ES5 suffisant). * Utile en pédagogie pour comprendre le fonctionnement bas niveau. | **Inconvénients :**   * Code beaucoup plus long et verbeux. * Risque d’erreurs dans les boucles imbriquées. * Performance légèrement inférieure pour les grands jeux de données (car pas d’optimisations natives comme filter/Set). |

|  |
| --- |
| **Solution retenue :**  Nous retenons l’approche de l’implémentation 1 avec le méthode modernes (forEach, filter, some, Set).  La raison est que ce code est plus maintenable, claire, performante et moins risque d’erreur |