

Fiche d’investigation fonctionnalité #3

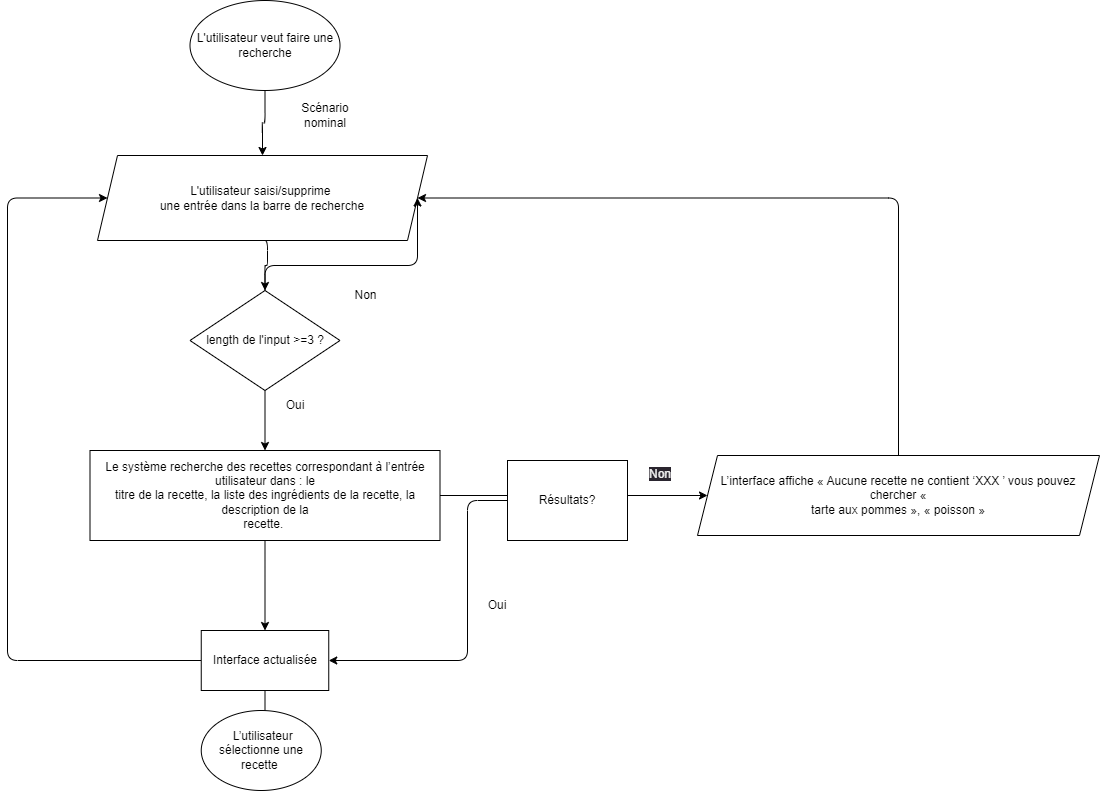
**Fiche d’investigation de fonctionnalité**

|  |  |
| --- | --- |
| **Fonctionnalité :** Filtrer les recettes dans l’interface utilisateur | **Fonctionnalité #3** |
| **Problématique :** Accéder rapidement à une recette correspondant à un besoin de l’utilisateur dans les recettes déjà reçues | |

|  |  |
| --- | --- |
| **Option 1 :** Utilisation des boucles natives (while, for...) | |
| **Avantages :**  Flexibilité : pratique pour effectuer des tâches simples ou complexes, telles que la recherche, le tri, la transformation de données, etc.  Compréhension intuitive : Les boucles while et for sont généralement faciles à comprendre, ce qui facilite la maintenance. | **Inconvénients :**  Risque d'erreurs : Les boucles natives peuvent être sujettes à des erreurs de programmation courantes telles que les boucles infinies (un problème où la boucle ne se termine jamais) ou les erreurs de décalage (un problème où l'index de boucle dépasse les limites d'un tableau). Ces erreurs peuvent être difficiles à repérer et à corriger.  Complexité potentielle : Les boucles natives peuvent rendre le code plus complexe, en particulier si elles sont imbriquées ou utilisées de manière inappropriée. Cela peut rendre le code plus difficile à déboguer et à entretenir. |
| Nombre de champs minimum à remplir dans la barre de recherche : 3 | |

|  |  |
| --- | --- |
| **Option 2 :** Programmation fonctionnelle avec les méthodes de l'objet array (foreach, filter, map, reduce) | |
| **Avantages :**  **Code plus lisible et déclaratif :** La programmation fonctionnelle avec les méthodes d'array rend souvent le code plus lisible en exprimant les opérations sous forme de fonctions de transformation ou de filtres. Cela permet de se concentrer sur ce que l'on veut faire plutôt que sur comment le faire.  **Réduction des effets secondaires** : La programmation fonctionnelle encourage l'écriture de fonctions pures, ce qui signifie que les fonctions n'ont pas d'effets secondaires et dépendent uniquement de leurs entrées. Cela peut rendre le code plus prévisible et plus facile à tester et à déboguer. | **Inconvénients :**  **Complexité accrue** : Bien que la programmation fonctionnelle puisse rendre le code plus lisible, elle peut également introduire de la complexité, en particulier lorsque plusieurs opérations sont enchaînées. Les erreurs peuvent être plus difficiles à diagnostiquer en raison de la chaîne d'opérations.  **Performance** : Dans certains cas, l'utilisation excessive de méthodes de programmation fonctionnelle peut entraîner une surcharge de performances par rapport à des boucles natives, en particulier pour de très grandes collections de données. Les opérations de carte et de réduction peuvent nécessiter plus de ressources que des boucles traditionnelles pour des tâches simples. |
| Nombre de champs minimum à remplir dans la barre de recherche : 3 | |

|  |
| --- |
| **Solution retenue :**  Après avoir fait une comparaison des 2 algorithmes via un benchmarking, l’option 2 utilisant un filter sera retenue car plus rapide. Source : <https://jsben.ch/coH0j> |

****