Fiche d’investigation de fonctionnalité

Fonctionnalité : Search	**Fonctionnalité #2**
Problématique : Afin de pouvoir retenir un maximum d’utilisateurs, nous cherchons à avoir une séquence de recherche la plus fluide et rapide possible.	

Option 1 : boucles natives

Dans cette option, nous avons utilisons la boucle native « for ». L’avantage de ce système c’est qu’il est rapide, car les boucles « for » s’executent en de manière synchrone. Le principal inconvénient : la boucle native « for » n’est pas très lisible lors de la lecture du code, elle est donc moins maintenable.

Avantages	**Inconvénients**
⊕ rapidité d’itération	⊖ Pas très lisible et donc moins maintenable

Pour un setup équivalent :

1258324.75 ops/s plus ou moins 2.28 %

Option 2 : méthodes de l'objet array

Dans cette option, nous avons utilisons la méthode de l’objet array « filter ». L’avantage de ce système c’est qu’il est plus facile à relire et donc plus maintenable, car la méthodde « filter » est plus descriptif. Le principal inconvénient : la méthode « filter » est moins performante sur de grande itération.

Avantages	**Inconvénients**
⊕ Code plus lisible est donc plus maintenable	⊖ faible performance sur de grande itération

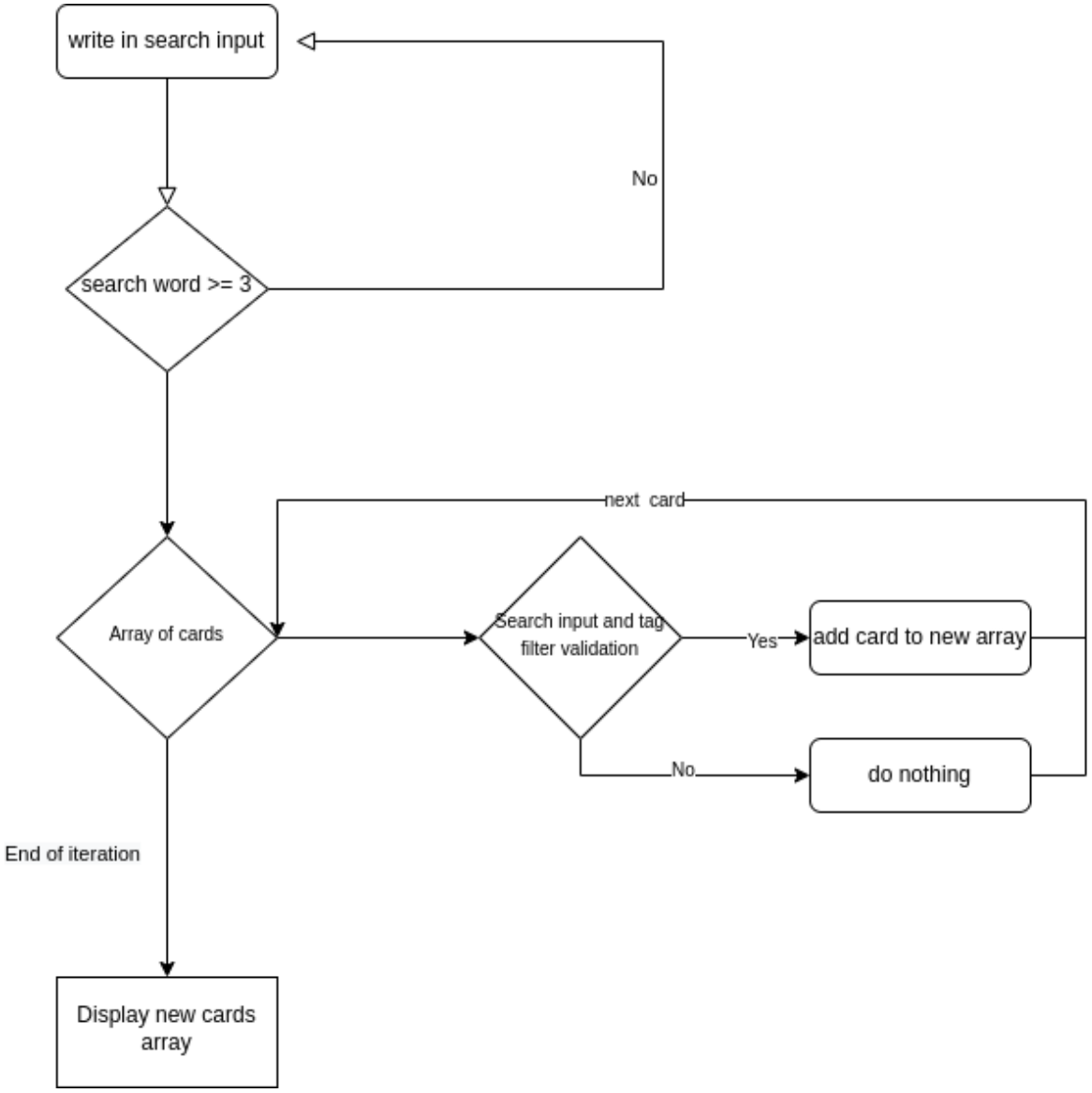
Pour un setup équivalent:

606425.47 ops/s plus ou moins 2.14 %

Solution retenue :

Nous avons donc retenu l'approche de la boucle native mais en gardant à l’esprit qu’il faut bien structurer le code afin qu’il soit lisible. La raison est que pour cette fonctionnalité la performance est privilegié.

Annexe


write in search input
No
search word >= 3
next card
Array of cards
Search input and tag filter validation
Yes
add card to new array
No
do nothing
End of iteration
Display new cards array