Les petits plats

Fiche d’investigation de fonctionnalité

Fonctionnalité	Filtrage de recettes selon divers critères
Problématique	Comment optimiser l’algorithme de filtrage des recettes (par nom, description, ingrédients, appareils, ustensiles) en comparant deux styles d'implémentation distincts (boucles for vs méthodes fonctionnelles), tout en conservant la même logique de filtrage.

Option 1 : Utilisation des boucles for (Approche impérative)

Description	Implémentation avec des boucles `for` pour chaque critère de filtrage, sans utiliser de méthodes fonctionnelles modernes sauf `includes` pour les recherches de sous-chaînes.
Avantages	⊕ **Simplicité du contrôle** : Chaque étape du filtrage est explicite et modifiable facilement. ⊕ **Compatibilité** : Fonctionne sur les anciennes versions de JavaScript. ⊕ **Débogage facilité** : Le flux est clair, facilitant l’ajout de points de contrôle.
Inconvénients	⊖ **Longueur du code** : Plus verbeux et moins lisible. ⊖ **Performance** : Moins performant pour les grandes quantités de données à cause des boucles imbriquées.
Nombre de méthodes utilisées	2 (`for`, `includes`)
Nombre de lignes de code	150+

Option 2 : Utilisation des méthodes fonctionnelles (filter, some, etc.)

Description	Approche fonctionnelle utilisant `filter`, `some`, `forEach`, et `includes` pour chaque critère de filtrage. Aucun `for` utilisé.
Avantages	⊕ **Lisibilité et concision :** Le code est plus compact et lisible. ⊕ **Modularité :** Les méthodes fonctionnelles permettent un code modulaire et plus maintenable. ⊕ **Optimisation :** Meilleure performance sur les grands tableaux grâce aux moteurs JavaScript modernes.
Inconvénients	⊖ **Moins de contrôle explicite :** Moins de flexibilité si un contrôle précis est nécessaire. ⊖ **Compatibilité :** Requiert un environnement moderne (ES5+).
Nombre de méthodes utilisées	4 (`filter`, `some`, `forEach`, `includes`)
Nombre de lignes de code	80+

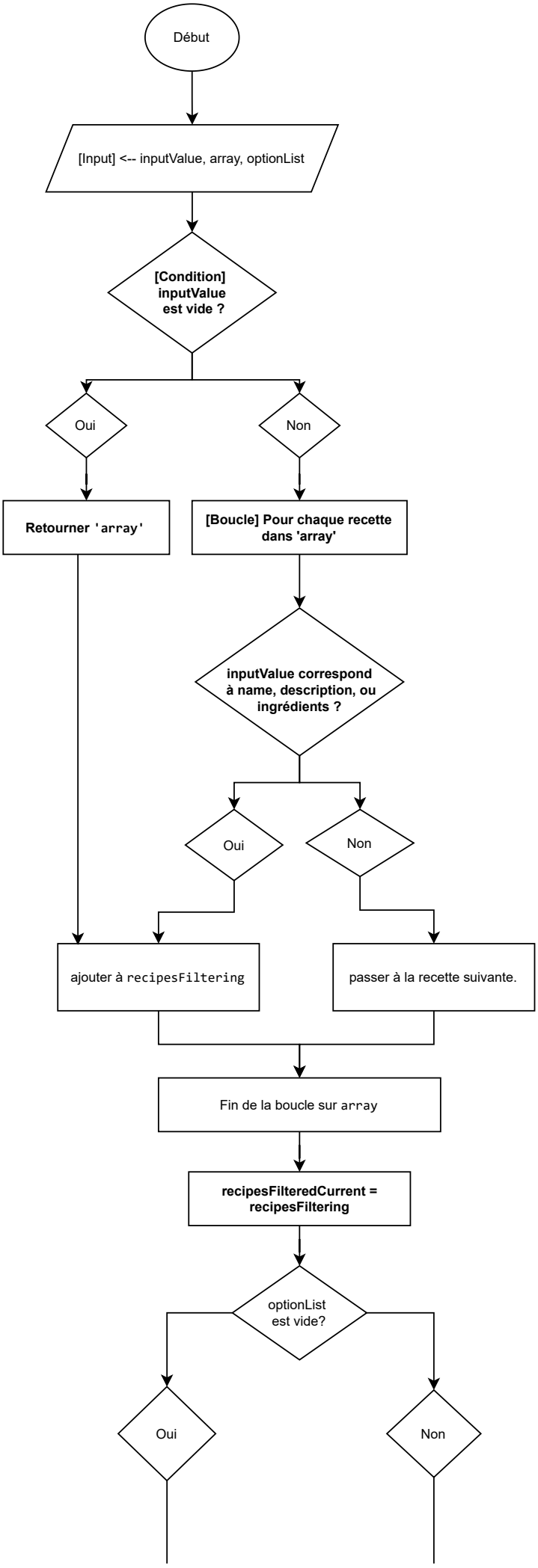
Comparaison des deux approches

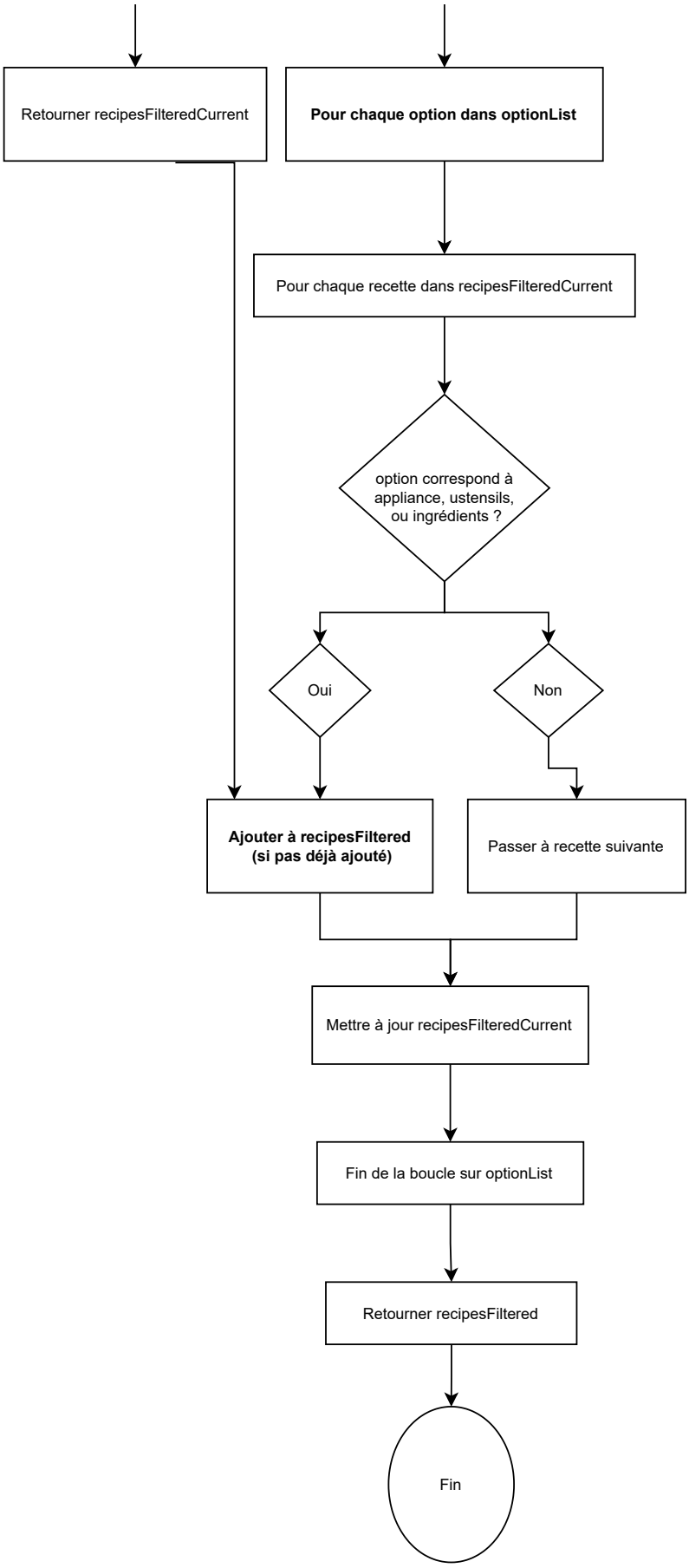
Critère	**Version 1 :** Boucles for	**Version 2 :** Méthodes fonctionnelles
Lisibilité et concision	Plus verbeux, moins lisible	Plus lisible et plus concis
Compatibilité	Compatible avec toutes les versions de JS	Nécessite ES5 ou plus récent
Performance potentielle	Moins performante avec de grands volumes	Optimisée pour de grands volumes
Contrôle explicite	Contrôle total sur chaque étape	Moins flexible avec les opérations standard

Solution retenue :

Nous avons retenu l'approche fonctionnelle pour sa lisibilité, sa modularité, et son optimisation des performances sur des tableaux volumineux, tout en tenant compte de la compatibilité avec notre environnement moderne.

Option 1 : Diagramme d’activité de l’approche avec des boucles for




Retourner recipesFilteredCurrent
Pour chaque option dans optionList
Pour chaque recette dans recipesFilteredCurrent
option correspond à appliance, ustensils, ou ingrédients ?
Oui
Non
Ajouter à recipesFiltered (si pas déjà ajouté)
Passer à recette suivante
Mettre à jour recipesFilteredCurrent
Fin de la boucle sur optionList
Retourner recipesFiltered
Fin

Option 2 : Diagramme d’activité de l’approche fonctionnelle (filter, some, forEach, etc.).