APPROCHE DE SOLUTION POUR LA LENTEUR DE L'APP BYPLUS

Problèmes / Solutions

1. Trop de composants rendus en même temps

- Utiliser lazy loading / Suspense pour ne charger que ce qui est visible.
- Fragmenter en composants plus petits et ne charger les données que dans le parent (puis distribuer via props)
- Virtualiser les listes (ex. `vue-virtual-scroller`) pour n'afficher que les éléments visibles.
- utiliser des pages pour certains composant avec des redirection

2. Propriétés réactives trop lourdes (watch et computed mal optimisés)

- Limiter les dépendances réactives des `computed`
- Sur les `watch`, cibler uniquement la valeur utile au lieu de `deep: true`
- Utiliser `shallowRef` ou `shallowReactive` pour éviter la réactivité profonde quand ce n'est pas nécessaire.

3. Boucles v-for volumineuses

- Centraliser les appels API dans le parent → passer les données via props
- Ajouter pagination ou infinite scroll
- Virtualiser la liste (n'afficher que les éléments visibles)

4. Side effect de reactive utilisé avec de grosse donnée (tableau, objet)

- Passer uniquement les données nécessaires en props
- Utiliser `markRaw()` ou `shallowRef()` pour éviter la réactivité profonde sur des objets massifs.
- Éviter de recréer les arrays/objets (`{...data}`) à chaque rendu → utiliser `computed` mémorisés.

5. Hooks mal placés

- Utiliser `onMounted()` pour ne déclencher l'API qu'une fois après montage
- Si données globales → les sortir dans un store (Pinia, Vuex) pour les charger une seule fois et les réutiliser.

6. Composant recréé inutilement

- Vérifier que la clé est stable (ex. `:key="id"` au lieu de `:key="Math.random()"`)

- Avec `<router-view>` → enlever la `:key` si elle n'est pas nécessaire, ou utiliser une clé basée uniquement sur le vrai changement attendu (`to.params.id`)

7. Watchers trop larges

- Surveiller une valeur spécifique
- Éviter `{ deep: true }` sauf cas indispensable

8. Effets déclenchés plusieurs fois

- Si besoin de garder le composant vivant → utiliser `<KeepAlive>` (surtout avec `<router-view>`)
- Dans `onBeforeRouteUpdate`, vérifier les conditions avant de relancer l'API

9. Pas de debounce/throttle

- Utiliser debounce ou throttle avec Lodash ou maison
- Utiliser `@change` au lieu de `@input` si tu veux appeler seulement quand l'utilisateur valide

10. Réactivité profonde involontaire

- Passer uniquement la valeur nécessaire (ex. `ref(filters.search)` au lieu de `reactive(filters)`).
- Utiliser `toRefs()` pour découper un objet réactif et ne réagir qu'à certaines parties.
- Si pas besoin de réactivité → passer en objet normal (`markRaw()`).

11. Trop de requêtes relancées inutilement

- Cache client-side pour éviter les refetch inutiles(vuex-persistedstate, que-query)