# Étape 3 : Recherche et filtres

## Objectif

Ajouter des fonctionnalités de recherche et filtrage à l'annuaire.

## Concepts pratiqués

- useState pour la gestion d'état
- Controlled inputs (inputs contrôlés)
- Filtrage avec filter()
- Événements onChange

---

## À créer

### 1. Composant `SearchBar.jsx`

Une barre de recherche qui :

- Est un input contrôlé (valeur dans le state)
- Appelle une fonction à chaque changement
- Affiche un bouton pour effacer la recherche

### 2. Composant `DepartmentFilter.jsx`

Un filtre par département qui :

- Affiche tous les départements disponibles
- Permet de sélectionner un département
- Affiche "Tous" par défaut

### 3. Mise à jour de `EmployeeList.jsx`

Intégrer :

- La barre de recherche
- Le filtre par département
- La logique de filtrage combinée

---

## Indices

▶ 💡 Input contrôlé avec useState

```
// L'input est "contrôlé" : sa valeur vient du state React
const [searchTerm, setSearchTerm] = useState('');

<input
```

```
  type="text"
  value={searchTerm}                    // La valeur vient du state
  onChange={(e) => setSearchTerm(e.target.value)}  // On met à jour le state
  placeholder="Rechercher..."
/>
```

**Pourquoi "contrôlé" ?**

- React contrôle la valeur de l'input
- On peut facilement réinitialiser la valeur
- On peut valider/transformer avant d'afficher

▶ 💡 Filtrer avec filter()

```
// filter() retourne un nouveau tableau avec les éléments qui passent le test
const filteredEmployees = employees.filter(employee => {
  // Recherche dans le nom
  const matchesSearch = employee.firstName
    .toLowerCase()
    .includes(searchTerm.toLowerCase());

  // Filtre par département
  const matchesDepartment =
    selectedDepartment === 'Tous' ||
    employee.department === selectedDepartment;

  // Doit passer les DEUX conditions
  return matchesSearch && matchesDepartment;
});
```

▶ 💡 Combiner plusieurs filtres

```
// On peut chaîner les conditions dans le filter
const filtered = employees.filter(emp => {
  // Condition 1 : recherche (si vide, tout passe)
  const matchesSearch = searchTerm === '' ||
    `${emp.firstName}
${emp.lastName}`.toLowerCase().includes(searchTerm.toLowerCase());

  // Condition 2 : département
  const matchesDept = department === 'Tous' || emp.department === department;

  return matchesSearch && matchesDept;
});
```

---

## Points d'attention

### ⚠ Input contrôlé vs non contrôlé

```
// ❌ Non contrôlé : React ne gère pas la valeur
<input type="text" />

// ☑ Contrôlé : React gère la valeur via le state
<input type="text" value={search} onChange={e => setSearch(e.target.value)} />
```

### ⚠ Recherche insensible à la casse

```
// Toujours convertir en minuscules pour comparer
const searchLower = searchTerm.toLowerCase();
const nameLower = employee.firstName.toLowerCase();
nameLower.includes(searchLower);
```

### ⚠ State dans le bon composant

La question : où mettre le state `searchTerm` et `selectedDepartment` ?

**Réponse** : Dans `EmployeeList`, car c'est lui qui doit filtrer les données.

Les composants `SearchBar` et `DepartmentFilter` reçoivent :

- La valeur actuelle (pour l'afficher)
- Une fonction de callback (pour la modifier)

---

## Critères de validation

- ☐ La recherche filtre en temps réel
- ☐ Le filtre département fonctionne
- ☐ Les deux filtres se combinent
- ☐ On peut réinitialiser la recherche
- ☐ Pas de lag perceptible