Bonnes pratiques

Bonnes pratiques React

Structure de fichiers

```
src/
├── components/          # Composants réutilisables
│   ├── Button/
│   │   ├── Button.jsx
│   │   └── Button.css
│   └── Modal/
├── pages/               # Composants de page (1 par route)
├── hooks/               # Custom hooks
├── contexts/            # Contextes React
├── services/            # Appels API
├── utils/               # Fonctions utilitaires
└── types/               # TypeScript types/interfaces
```

Règles à respecter

- **Un composant = une responsabilité** — si un composant est trop gros, le découper
- **Extraire la logique** dans des custom hooks (`useForm`, `useFetch`...)
- **Remonter l'état** (lift state up) au plus proche ancêtre commun qui en a besoin
- **Toujours mettre une key** sur les éléments d'une liste — idéalement un ID, pas l'index
- **Éviter l'état dérivé** — ne pas stocker dans le state ce qui peut être calculé

Anti-patterns courants

```
// ❌ Synchroniser deux états (état dérivé)
const [items, setItems] = useState([]);
const [count, setCount] = useState(0);
// Chaque ajout doit mettre à jour les deux

// ✅ Calculer le dérivé
```

```
const [items, setItems] = useState([]);
const count = items.length; // Calculé, pas stocké

// ❌ useEffect pour synchroniser des states
useEffect(() => setFullName(first + " " + last), [first, last]);

// ✅ Variable calculée
const fullName = first + " " + last;

// ❌ Key = index (problèmes lors de réordonnancement)
{items.map((item, i) => <Item key={i} />)}

// ✅ Key = ID stable
{items.map(item => <Item key={item.id} />)}
```

Revision #1
Created 29 March 2026 12:57:40 by Hugo
Updated 29 March 2026 12:57:40 by Hugo