# Optimisations

## Optimisations de performance

### React.memo — éviter les re-renders inutiles

```
// L'enfant se re-rend seulement si ses props changent
const ListItem = React.memo(function ListItem({ item, onDelete }) {
  return (
    <li>
      {item.nom}
      <button onClick={() => onDelete(item.id)}>Supprimer</button>
    </li>
  );
});
```

### Code splitting — lazy loading

```
import { lazy, Suspense } from 'react';

// Chargé seulement quand la route est visitée
const Dashboard = lazy(() => import('./pages/Dashboard'));
const Settings = lazy(() => import('./pages/Settings'));

function App() {
  return (
    <Suspense fallback={<Spinner />}>
      <Routes>
        <Route path="/dashboard" element={<Dashboard />} />
        <Route path="/settings" element={<Settings />} />
      </Routes>
    </Suspense>
  );
}
```

# Virtualisation des listes

Pour des listes de centaines/milliers d'éléments, n'afficher que ce qui est visible :

```
npm install @tanstack/react-virtual

import { useVirtualizer } from '@tanstack/react-virtual';

function GrandeListe({ items }) {
  const parentRef = useRef(null);

  const virtualizer = useVirtualizer({
    count: items.length,
    getScrollElement: () => parentRef.current,
    estimateSize: () => 50,
  });

  return (
    <div ref={parentRef} style={{ height: '400px', overflow: 'auto' }}>
      <div style={{ height: virtualizer.getTotalSize() }}>
        {virtualizer.getVirtualItems().map(vItem => (
          <div key={vItem.index} style={{ transform: `translateY(${vItem.start}px)` }}>
            {items[vItem.index].nom}
          </div>
        ))}
      </div>
    </div>
  );
}
```

---

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