Exercice 10 - Mini Application Complète

Objectif

Synthèse de tous les concepts du Module 2 dans une application de gestion de tâches avancée.

Concepts combinés

- useState, useEffect, useReducer, useRef
- Custom hooks
- Context API
- Fetch API / Simulation d'API
- Optimisation (memo, useCallback)
- Persistance localStorage

Énoncé

Créer une application **TaskManager** avec :

1. **Liste de tâches** avec statut (todo, in-progress, done)
2. **Filtres et recherche**
3. **Ajout/Édition/Suppression**
4. **Drag & Drop** pour changer le statut (bonus)
5. **Persistance** dans localStorage
6. **Thème** clair/sombre
7. **Stats** (nombre de tâches par statut)

Correction

Structure des fichiers

```
src/
├── contexts/
│   └── ThemeContext.jsx
├── hooks/
│   ├── useLocalStorage.js
│   └── useTasks.js
├── components/
│   ├── TaskBoard.jsx
│   ├── TaskColumn.jsx
│   ├── TaskCard.jsx
│   ├── TaskForm.jsx
│   ├── TaskStats.jsx
│   └── ThemeToggle.jsx
├── App.jsx
└── index.css
```

1. Hook useTasks (useReducer + localStorage)

```
// hooks/useTasks.js
import { useReducer, useEffect } from 'react';

const initialState = {
  tasks: [],
  filter: 'all', // all, todo, in-progress, done
  searchTerm: ''
};

function tasksReducer(state, action) {
  switch (action.type) {
    case 'LOAD_TASKS':
      return { ...state, tasks: action.payload };

    case 'ADD_TASK':
      return {
        ...state,
        tasks: [...state.tasks, {
          id: Date.now(),
          title: action.payload.title,
          description: action.payload.description || '',
          status: 'todo',
          priority: action.payload.priority || 'medium',
          createdAt: new Date().toISOString()
        }]
      };

    case 'UPDATE_TASK':
      return {
        ...state,
        tasks: state.tasks.map(task =>
          task.id === action.payload.id
            ? { ...task, ...action.payload.updates }
            : task
        )
      };

    case 'DELETE_TASK':
      return {
        ...state,
        tasks: state.tasks.filter(task => task.id !== action.payload)
      };

    case 'MOVE_TASK':
      return {
        ...state,
        tasks: state.tasks.map(task =>
          task.id === action.payload.id
            ? { ...task, status: action.payload.status }
```

```
            : task
        )
      };

    case 'SET_FILTER':
      return { ...state, filter: action.payload };

    case 'SET_SEARCH':
      return { ...state, searchTerm: action.payload };

    default:
      return state;
  }
}

export function useTasks() {
  const [state, dispatch] = useReducer(tasksReducer, initialState);

  // Charger depuis localStorage au montage
  useEffect(() => {
    const saved = localStorage.getItem('tasks');
    if (saved) {
      dispatch({ type: 'LOAD_TASKS', payload: JSON.parse(saved) });
    }
  }, []);

  // Sauvegarder dans localStorage à chaque changement
  useEffect(() => {
    localStorage.setItem('tasks', JSON.stringify(state.tasks));
  }, [state.tasks]);

  // Filtrer les tâches
  const filteredTasks = state.tasks.filter(task => {
    const matchesFilter = state.filter === 'all' || task.status === state.filter;
    const matchesSearch =
task.title.toLowerCase().includes(state.searchTerm.toLowerCase());
    return matchesFilter && matchesSearch;
  });

  // Stats
  const stats = {
    total: state.tasks.length,
    todo: state.tasks.filter(t => t.status === 'todo').length,
    inProgress: state.tasks.filter(t => t.status === 'in-progress').length,
    done: state.tasks.filter(t => t.status === 'done').length
  };

  return {
    tasks: filteredTasks,
    allTasks: state.tasks,
    filter: state.filter,
    searchTerm: state.searchTerm,
    stats,
    dispatch
```

```
  };
}
```

2. Composant TaskBoard

```jsx
// components/TaskBoard.jsx
import { useTasks } from '../hooks/useTasks';
import TaskColumn from './TaskColumn';
import TaskForm from './TaskForm';
import TaskStats from './TaskStats';
import './TaskBoard.css';

function TaskBoard() {
  const { tasks, allTasks, filter, searchTerm, stats, dispatch } = useTasks();

  const columns = [
    { id: 'todo', title: 'À faire', color: '#f59e0b' },
    { id: 'in-progress', title: 'En cours', color: '#3b82f6' },
    { id: 'done', title: 'Terminé', color: '#10b981' }
  ];

  function handleAddTask(taskData) {
    dispatch({ type: 'ADD_TASK', payload: taskData });
  }

  function handleMoveTask(taskId, newStatus) {
    dispatch({ type: 'MOVE_TASK', payload: { id: taskId, status: newStatus } });
  }

  function handleDeleteTask(taskId) {
    dispatch({ type: 'DELETE_TASK', payload: taskId });
  }

  function handleUpdateTask(taskId, updates) {
    dispatch({ type: 'UPDATE_TASK', payload: { id: taskId, updates } });
  }

  return (
    <div className="task-board">
      {/* Header */}
      <header className="board-header">
        <h1>Task Manager</h1>
        <TaskStats stats={stats} />
      </header>

      {/* Filtres */}
      <div className="board-filters">
        <input
          type="text"
          placeholder="Rechercher..."
          value={searchTerm}
```

```
          onChange={e => dispatch({ type: 'SET_SEARCH', payload: e.target.value
})}
          className="search-input"
        />
        <select
          value={filter}
          onChange={e => dispatch({ type: 'SET_FILTER', payload: e.target.value
})}
          className="filter-select"
        >
          <option value="all">Toutes</option>
          <option value="todo">À faire</option>
          <option value="in-progress">En cours</option>
          <option value="done">Terminées</option>
        </select>
      </div>

      {/* Formulaire d'ajout */}
      <TaskForm onSubmit={handleAddTask} />

      {/* Colonnes */}
      <div className="board-columns">
        {columns.map(column => (
          <TaskColumn
            key={column.id}
            column={column}
            tasks={tasks.filter(t => t.status === column.id)}
            onMoveTask={handleMoveTask}
            onDeleteTask={handleDeleteTask}
            onUpdateTask={handleUpdateTask}
          />
        ))}
      </div>
    </div>
  );
}

export default TaskBoard;
```

3. Composant TaskColumn

```
// components/TaskColumn.jsx
import { memo } from 'react';
import TaskCard from './TaskCard';
import './TaskColumn.css';

const TaskColumn = memo(function TaskColumn({
  column,
  tasks,
  onMoveTask,
  onDeleteTask,
```

```
  onUpdateTask
}) {
  return (
    <div className="task-column">
      <div
        className="column-header"
        style={{ borderColor: column.color }}
      >
        <h2>{column.title}</h2>
        <span className="task-count">{tasks.length}</span>
      </div>

      <div className="column-content">
        {tasks.length === 0 ? (
          <p className="empty-message">Aucune tâche</p>
        ) : (
          tasks.map(task => (
            <TaskCard
              key={task.id}
              task={task}
              onMove={onMoveTask}
              onDelete={onDeleteTask}
              onUpdate={onUpdateTask}
            />
          ))
        )}
      </div>
    </div>
  );
});

export default TaskColumn;
```

4. Composant TaskCard

```
// components/TaskCard.jsx
import { memo, useState } from 'react';
import './TaskCard.css';

const TaskCard = memo(function TaskCard({ task, onMove, onDelete, onUpdate }) {
  const [isEditing, setIsEditing] = useState(false);
  const [editTitle, setEditTitle] = useState(task.title);

  const priorities = {
    low: { label: 'Basse', color: '#22c55e' },
    medium: { label: 'Moyenne', color: '#f59e0b' },
    high: { label: 'Haute', color: '#ef4444' }
  };

  function handleSave() {
    onUpdate(task.id, { title: editTitle });
```

```
    setIsEditing(false);
  }

  function handleStatusChange(e) {
    onMove(task.id, e.target.value);
  }

  return (
    <div className={`task-card priority-${task.priority}`}>
      {isEditing ? (
        <div className="edit-mode">
          <input
            type="text"
            value={editTitle}
            onChange={e => setEditTitle(e.target.value)}
            autoFocus
          />
          <div className="edit-actions">
            <button onClick={handleSave} className="btn-save">✓</button>
            <button onClick={() => setIsEditing(false)} className="btn-
cancel">✕</button>
          </div>
        </div>
      ) : (
        <>
          <div className="task-header">
            <h3 onClick={() => setIsEditing(true)}>{task.title}</h3>
            <span
              className="priority-badge"
              style={{ backgroundColor: priorities[task.priority].color }}
            >
              {priorities[task.priority].label}
            </span>
          </div>

          {task.description && (
            <p className="task-description">{task.description}</p>
          )}

          <div className="task-footer">
            <select value={task.status} onChange={handleStatusChange}>
              <option value="todo">À faire</option>
              <option value="in-progress">En cours</option>
              <option value="done">Terminé</option>
            </select>

            <button
              onClick={() => onDelete(task.id)}
              className="btn-delete"
              title="Supprimer"
            >
              🗑
            </button>
          </div>
```

```
        </>
      )}
    </div>
  );
});

export default TaskCard;
```

5. Composant TaskForm

```
// components/TaskForm.jsx
import { useState } from 'react';
import './TaskForm.css';

function TaskForm({ onSubmit }) {
  const [isOpen, setIsOpen] = useState(false);
  const [title, setTitle] = useState('');
  const [description, setDescription] = useState('');
  const [priority, setPriority] = useState('medium');

  function handleSubmit(e) {
    e.preventDefault();
    if (!title.trim()) return;

    onSubmit({ title, description, priority });
    setTitle('');
    setDescription('');
    setPriority('medium');
    setIsOpen(false);
  }

  if (!isOpen) {
    return (
      <button onClick={() => setIsOpen(true)} className="add-task-btn">
        + Nouvelle tâche
      </button>
    );
  }

  return (
    <form onSubmit={handleSubmit} className="task-form">
      <input
        type="text"
        placeholder="Titre de la tâche"
        value={title}
        onChange={e => setTitle(e.target.value)}
        autoFocus
        required
      />

      <textarea
```

```
          placeholder="Description (optionnel)"
          value={description}
          onChange={e => setDescription(e.target.value)}
          rows={2}
        />

        <div className="form-row">
          <select value={priority} onChange={e => setPriority(e.target.value)}>
            <option value="low">Priorité basse</option>
            <option value="medium">Priorité moyenne</option>
            <option value="high">Priorité haute</option>
          </select>

          <div className="form-actions">
            <button type="button" onClick={() => setIsOpen(false)}>
              Annuler
            </button>
            <button type="submit" className="btn-primary">
              Ajouter
            </button>
          </div>
        </div>
      </form>
    );
}

export default TaskForm;
```

6. Composant TaskStats

```
// components/TaskStats.jsx
import { memo } from 'react';
import './TaskStats.css';

const TaskStats = memo(function TaskStats({ stats }) {
  const completion = stats.total > 0
    ? Math.round((stats.done / stats.total) * 100)
    : 0;

  return (
    <div className="task-stats">
      <div className="stat">
        <span className="stat-value">{stats.total}</span>
        <span className="stat-label">Total</span>
      </div>
      <div className="stat todo">
        <span className="stat-value">{stats.todo}</span>
        <span className="stat-label">À faire</span>
      </div>
      <div className="stat in-progress">
        <span className="stat-value">{stats.inProgress}</span>
```

```
          <span className="stat-label">En cours</span>
        </div>
        <div className="stat done">
          <span className="stat-value">{stats.done}</span>
          <span className="stat-label">Terminé</span>
        </div>
        <div className="stat completion">
          <span className="stat-value">{completion}%</span>
          <span className="stat-label">Complété</span>
        </div>
      </div>
    );
  });

  export default TaskStats;
```

Récapitulatif des concepts utilisés

Concept	Utilisation
`useReducer`	Gestion complexe des tâches
`useEffect`	Chargement/sauvegarde localStorage
`useState`	États locaux (formulaires, édition)
`useCallback`	Stabiliser les handlers (optionnel)
`memo`	Éviter re-renders inutiles
Context	Thème global
Custom Hook	`useTasks`