Cours MERN - Semaine 7

Maîtriser les Hooks Essentiels React

`useState`, `useEffect` et `useReducer`

Abdelweheb GUEDDES & Mohamed Ben Jazia / Ecole Polytechnique Sousse

8 novembre 2025

Table des matières

1 Objectifs Pédagogiques **2**

2 Partie 1 : Concepts Essentiels (30 min) **2**
2.1 Hook 1 : `useState` - Ajouter de la mémoire . . . 2
2.1.1 L'Immutabilité : La Règle d'Or . . . 2
2.2 Hook 2 : `useEffect` - Les effets de bord . . . 3
2.3 Hook 3 : `useReducer` - État complexe . . . 3

3 Partie 2 : Projets Pratiques (2h30) **5**
3.1 Projet 1 : Gestionnaire de Tâches avec useReducer (45 min) . . . 5
3.1.1 Étape 1 : Créer le Reducer . . . 5
3.1.2 Étape 2 : Le Composant Principal . . . 6
3.2 Projet 2 : Liste de Courses avec Timer (45 min) . . . 11
3.2.1 Code Complet . . . 11
3.3 Projet 3 : Blog Interactif avec Tri et Recherche (60 min) . . . 18
3.3.1 Code Complet de l'Application . . . 18

4 Récapitulatif et Concepts Clés **26**

5 Travail à Rendre **27**

6 Ressources **27**
6.1 Documentation Officielle . . . 27
6.2 Outils . . . 28

1 Objectifs Pédagogiques

L'objectif central de cette séance est de maîtriser les trois hooks fondamentaux de React à travers des projets pratiques concrets.

À la fin de cette séance, vous serez capable de :

— **Utiliser** `useState` pour gérer l'état local simple et complexe
— **Maîtriser** `useEffect` pour les effets de bord (API, timers, localStorage)
— **Implémenter** `useReducer` pour organiser la logique d'état complexe
— **Appliquer** l'immutabilité correctement dans toutes les mises à jour d'état
— **Créer** des applications interactives complètes avec persistance de données

2 Partie 1 : Concepts Essentiels (30 min)

2.1 Hook 1 : `useState` - Ajouter de la mémoire

`useState` permet d'ajouter de l'état à un composant fonctionnel. React conserve cet état entre les rendus.

Syntaxe de base :

```
const [state, setState] = useState(initialValue);

// Exemples
const [count, setCount] = useState(0);
const [name, setName] = useState('');
const [items, setItems] = useState([]);
const [user, setUser] = useState({ name: '', age: 0 });
```

2.1.1 L'Immutabilité : La Règle d'Or

En React, l'état ne doit **jamais** être modifié directement.

INCORRECT - Mutation directe :

```
// Objets
user.name = "Bob";
setUser(user);  // React ne détecte pas le changement

// Tableaux
items.push(newItem);
setItems(items);  // React ne détecte pas le changement
```

CORRECT - Immutabilité :

```
// Objets
setUser({...user, name: "Bob"});  // Nouvel objet

// Tableaux
setItems([...items, newItem]);  // Nouveau tableau
```

Opérations immutables sur les tableaux :

Opération	Méthode immutable
Ajouter	`[...arr, newItem]`
Supprimer	`arr.filter((_,i) => i !== index)`
Modifier	`arr.map((item,i) => i === index ? newItem : item)`

2.2 Hook 2 : `useEffect` - Les effets de bord

`useEffect` permet d'exécuter du code après le rendu pour synchroniser avec des systèmes externes.

Syntaxe :

```
useEffect(() => {
    // Code à exécuter

    return () => {
        // Nettoyage (optionnel)
    };
}, [dependencies]);
```

Les trois cas d'usage :

1. **Une seule fois au montage :** `useEffect(() => {...}, [])`
2. **Quand une valeur change :** `useEffect(() => {...}, [value])`
3. **À chaque rendu :** `useEffect(() => {...})` (rarement utilisé)

Exemples courants :

— Charger des données depuis une API
— Créer un timer ou intervalle
— Sauvegarder dans localStorage
— S'abonner à des événements

2.3 Hook 3 : `useReducer` - État complexe

Quand l'état devient complexe (plusieurs sous-valeurs liées), `useReducer` offre une meilleure organisation.

Syntaxe :

```
const [state, dispatch] = useReducer(reducer, initialState);

// Fonction reducer
function reducer(state, action) {
    switch (action.type) {
        case 'ACTION_NAME':
            return {...state, /* modifications */};
```

```
        default:
            return state;
    }
}

// Utilisation
dispatch({ type: 'ACTION_NAME', payload: value });
```

Quand utiliser useReducer ?

— Plusieurs sous-valeurs d'état liées
— Logique de mise à jour complexe
— Transitions d'état multiples

3 Partie 2 : Projets Pratiques (2h30)

3.1 Projet 1 : Gestionnaire de Tâches avec useReducer (45 min)

Objectif : Créer une Todo List complète utilisant `useReducer` pour gérer l'état.

Fonctionnalités :

- Ajouter des tâches
- Marquer comme complétées
- Supprimer des tâches
- Filtrer (toutes / actives / complétées)
- Persistance avec localStorage

3.1.1 Étape 1 : Créer le Reducer

```
export const initialState = {
    todos: [],
    filter: 'all' // 'all', 'active', 'completed'
};

export function todoReducer(state, action) {
    switch (action.type) {
        case 'ADD_TODO':
            return {
                ...state,
                todos: [
                    {
                        id: Date.now(),
                        text: action.payload,
                        completed: false,
                        createdAt: new Date().toISOString()
                    },
                    ...state.todos
                ]
            };

        case 'TOGGLE_TODO':
            return {
                ...state,
                todos: state.todos.map(todo =>
                    todo.id === action.payload
                        ? {...todo, completed: !todo.completed}
                        : todo
                )
            };

        case 'DELETE_TODO':
            return {
                ...state,
                todos: state.todos.filter(todo =>
```

```
                    todo.id !== action.payload
                )
            };

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

        case 'LOAD_TODOS':
            return {
                ...state,
                todos: action.payload
            };

        default:
            return state;
    }
}
```

Listing 1 – src/reducers/todoReducer.js

3.1.2 Étape 2 : Le Composant Principal

```
import { useReducer, useState, useEffect } from 'react';
import { todoReducer, initialState } from './reducers/todoReducer';

function App() {
    const [state, dispatch] = useReducer(todoReducer, initialState);
    const [input, setInput] = useState('');

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

    // Sauvegarder dans localStorage à chaque modification
    useEffect(() => {
        if (state.todos.length > 0) {
            localStorage.setItem('todos', JSON.stringify(state.todos)
    );
        }
    }, [state.todos]);

```

```
    const addTodo = () => {
        if (input.trim()) {
            dispatch({ type: 'ADD_TODO', payload: input });
            setInput('');
        }
    };

    // Filtrer les todos
    const filteredTodos = state.todos.filter(todo => {
        if (state.filter === 'active') return !todo.completed;
        if (state.filter === 'completed') return todo.completed;
        return true;
    });

    return (
        <div style={{
            maxWidth: '600px',
            margin: '50px auto',
            padding: '20px',
            fontFamily: 'Arial, sans-serif'
        }}>
            <h1 style={{ textAlign: 'center', color: '#2c3e50' }}>
                Ma Todo List
            </h1>

            {/* Formulaire d'ajout */}
            <div style={{
                display: 'flex',
                gap: '10px',
                marginBottom: '20px'
            }}>
                <input
                    type="text"
                    value={input}
                    onChange={(e) => setInput(e.target.value)}
                    onKeyPress={(e) => e.key === 'Enter' && addTodo()
}
                    placeholder="Nouvelle tâche..."
                    style={{
                        flex: 1,
                        padding: '12px',
                        fontSize: '16px',
                        border: '2px solid #ddd',
                        borderRadius: '5px'
                    }}
                />
                <button
                    onClick={addTodo}
                    style={{
                        padding: '12px 24px',
                        backgroundColor: '#27ae60',
```

```
                        color: 'white',
                        border: 'none',
                        borderRadius: '5px',
                        cursor: 'pointer',
                        fontSize: '16px',
                        fontWeight: 'bold'
                    }}
                >
                    Ajouter
                </button>
            </div>

            {/* Filtres */}
            <div style={{
                display: 'flex',
                gap: '10px',
                marginBottom: '20px',
                justifyContent: 'center'
            }}>
                {['all', 'active', 'completed'].map(filter => (
                    <button
                        key={filter}
                        onClick={() => dispatch({
                            type: 'SET_FILTER',
                            payload: filter
                        })}
                        style={{
                            padding: '8px 16px',
                            backgroundColor: state.filter === filter
                                ? '#3498db'
                                : '#ecf0f1',
                            color: state.filter === filter
                                ? 'white'
                                : '#2c3e50',
                            border: 'none',
                            borderRadius: '5px',
                            cursor: 'pointer',
                            fontWeight: state.filter === filter
                                ? 'bold'
                                : 'normal'
                        }}
                    >
                        {filter === 'all' ? 'Toutes' :
                         filter === 'active' ? 'Actives' : 'Complété
es'}
                    </button>
                ))}
            </div>

            {/* Liste des tâches */}
            <div>
```

```
                {filteredTodos.length === 0 ? (
                    <p style={{
                        textAlign: 'center',
                        color: '#95a5a6',
                        padding: '40px'
                    }}>
                        Aucune tâche à afficher
                    </p>
                ) : (
                    filteredTodos.map(todo => (
                        <div
                            key={todo.id}
                            style={{
                                display: 'flex',
                                alignItems: 'center',
                                padding: '15px',
                                marginBottom: '10px',
                                backgroundColor: todo.completed
                                    ? '#d5f4e6'
                                    : 'white',
                                border: '1px solid #ddd',
                                borderRadius: '5px'
                            }}
                        >
                            <input
                                type="checkbox"
                                checked={todo.completed}
                                onChange={() => dispatch({
                                    type: 'TOGGLE_TODO',
                                    payload: todo.id
                                })}
                                style={{
                                    marginRight: '15px',
                                    width: '20px',
                                    height: '20px',
                                    cursor: 'pointer'
                                }}
                            />
                            <span style={{
                                flex: 1,
                                fontSize: '16px',
                                textDecoration: todo.completed
                                    ? 'line-through'
                                    : 'none',
                                color: todo.completed
                                    ? '#95a5a6'
                                    : '#2c3e50'
                            }}>
                                {todo.text}
                            </span>
                            <button
```

```
                                onClick={() => dispatch({
                                    type: 'DELETE_TODO',
                                    payload: todo.id
                                })}
                                style={{
                                    padding: '6px 12px',
                                    backgroundColor: '#e74c3c',
                                    color: 'white',
                                    border: 'none',
                                    borderRadius: '3px',
                                    cursor: 'pointer'
                                }}
                            >
                                Supprimer
                            </button>
                        </div>
                    ))
                )}
            </div>

            {/* Statistiques */}
            <div style={{
                marginTop: '20px',
                textAlign: 'center',
                padding: '15px',
                backgroundColor: '#ecf0f1',
                borderRadius: '5px'
            }}>
                <strong>Total :</strong> {state.todos.length} |
                <strong> Complétées :</strong> {
                    state.todos.filter(t => t.completed).length
                }
            </div>
        </div>
    );
}

export default App;
```

Listing 2 – src/App.jsx

Points clés de ce projet :

- **useReducer** centralise toute la logique d'état
- **useEffect** gère la persistance localStorage
- **useState** pour l'input du formulaire
- **Immutabilité** respectée dans tous les cas du reducer

3.2 Projet 2 : Liste de Courses avec Timer (45 min)

Objectif : Créer une liste de courses avec un timer Pomodoro intégré.

Compétences :

- useState avec différents types
- useEffect avec nettoyage (timer)
- Persistance localStorage
- Gestion de formulaire

3.2.1 Code Complet

```
import { useState, useEffect } from 'react';

function ShoppingListApp() {
    // Etats pour la liste
    const [items, setItems] = useState([]);
    const [inputValue, setInputValue] = useState('');
    const [inputPrice, setInputPrice] = useState('');

    // Etats pour le timer Pomodoro
    const [minutes, setMinutes] = useState(25);
    const [seconds, setSeconds] = useState(0);
    const [isActive, setIsActive] = useState(false);

    // Charger la liste depuis localStorage
    useEffect(() => {
        const saved = localStorage.getItem('shopping-list');
        if (saved) {
            setItems(JSON.parse(saved));
        }
    }, []);

    // Sauvegarder la liste
    useEffect(() => {
        localStorage.setItem('shopping-list', JSON.stringify(items));
    }, [items]);

    // Timer Pomodoro
    useEffect(() => {
        let interval = null;

        if (isActive) {
            interval = setInterval(() => {
                if (seconds === 0) {
                    if (minutes === 0) {
                        setIsActive(false);
                        alert('Temps de course écoulé !');
                    } else {
                        setMinutes(minutes - 1);
                        setSeconds(59);
```

```
                    }
                } else {
                    setSeconds(seconds - 1);
                }
            }, 1000);
        }

        // Nettoyage : arrêter l'intervalle
        return () => {
            if (interval) clearInterval(interval);
        };
    }, [isActive, minutes, seconds]);

    const addItem = () => {
        if (inputValue.trim()) {
            setItems([
                ...items,
                {
                    id: Date.now(),
                    name: inputValue,
                    price: inputPrice ? parseFloat(inputPrice) : 0,
                    bought: false
                }
            ]);
            setInputValue('');
            setInputPrice('');
        }
    };

    const toggleBought = (id) => {
        setItems(items.map(item =>
            item.id === id
                ? {...item, bought: !item.bought}
                : item
        ));
    };

    const deleteItem = (id) => {
        setItems(items.filter(item => item.id !== id));
    };

    const resetTimer = () => {
        setIsActive(false);
        setMinutes(25);
        setSeconds(0);
    };

    const totalPrice = items.reduce((sum, item) => sum + item.price,
0);
    const boughtCount = items.filter(item => item.bought).length;

```

```
    return (
        <div style={{
            maxWidth: '700px',
            margin: '50px auto',
            padding: '20px'
        }}>
            <h1 style={{ textAlign: 'center' }}>
                Liste de Courses
            </h1>

            {/* Timer Pomodoro */}
            <div style={{
                backgroundColor: '#34495e',
                color: 'white',
                padding: '20px',
                borderRadius: '10px',
                marginBottom: '30px',
                textAlign: 'center'
            }}>
                <h3>Timer de Course</h3>
                <div style={{
                    fontSize: '48px',
                    fontWeight: 'bold',
                    margin: '20px 0',
                    color: isActive ? '#e74c3c' : 'white'
                }}>
                    {String(minutes).padStart(2, '0')}:
                    {String(seconds).padStart(2, '0')}
                </div>
                <div style={{ display: 'flex', gap: '10px',
justifyContent: 'center' }}>
                    <button
                        onClick={() => setIsActive(!isActive)}
                        style={{
                            padding: '10px 20px',
                            backgroundColor: isActive ? '#f39c12' : '
#27ae60',
                            color: 'white',
                            border: 'none',
                            borderRadius: '5px',
                            cursor: 'pointer',
                            fontSize: '16px'
                        }}
                    >
                        {isActive ? 'Pause' : 'Démarrer'}
                    </button>
                    <button
                        onClick={resetTimer}
                        style={{
                            padding: '10px 20px',
                            backgroundColor: '#95a5a6',
```

```
                                color: 'white',
                                border: 'none',
                                borderRadius: '5px',
                                cursor: 'pointer',
                                fontSize: '16px'
                            }}
                        >
                            Reset
                        </button>
                    </div>
                </div>

                {/* Formulaire d'ajout */}
                <div style={{
                    backgroundColor: '#ecf0f1',
                    padding: '20px',
                    borderRadius: '10px',
                    marginBottom: '20px'
                }}>
                    <div style={{ display: 'flex', gap: '10px',
    marginBottom: '10px' }}>
                        <input
                            type="text"
                            value={inputValue}
                            onChange={(e) => setInputValue(e.target.value
    )}
                            onKeyPress={(e) => e.key === 'Enter' &&
    addItem()}
                            placeholder="Nom de l'article..."
                            style={{
                                flex: 2,
                                padding: '10px',
                                fontSize: '16px',
                                border: '1px solid #bdc3c7',
                                borderRadius: '5px'
                            }}
                        />
                        <input
                            type="number"
                            value={inputPrice}
                            onChange={(e) => setInputPrice(e.target.value
    )}
                            onKeyPress={(e) => e.key === 'Enter' &&
    addItem()}
                            placeholder="Prix..."
                            step="0.01"
                            style={{
                                flex: 1,
                                padding: '10px',
                                fontSize: '16px',
                                border: '1px solid #bdc3c7',
```

```
                            borderRadius: '5px'
                        }}
                    />
                    <button
                        onClick={addItem}
                        style={{
                            padding: '10px 20px',
                            backgroundColor: '#3498db',
                            color: 'white',
                            border: 'none',
                            borderRadius: '5px',
                            cursor: 'pointer',
                            fontSize: '16px'
                        }}
                    >
                        Ajouter
                    </button>
                </div>
            </div>

            {/* Liste des articles */}
            <div>
                {items.map(item => (
                    <div
                        key={item.id}
                        style={{
                            display: 'flex',
                            alignItems: 'center',
                            padding: '15px',
                            marginBottom: '10px',
                            backgroundColor: item.bought ? '#d5f4e6' : 'white',
                            border: '1px solid #ddd',
                            borderRadius: '5px'
                        }}
                    >
                        <input
                            type="checkbox"
                            checked={item.bought}
                            onChange={() => toggleBought(item.id)}
                            style={{
                                marginRight: '15px',
                                width: '20px',
                                height: '20px'
                            }}
                        />
                        <span style={{
                            flex: 1,
                            textDecoration: item.bought ? 'line-through' : 'none',
                            color: item.bought ? '#95a5a6' : '#2c3e50
```

```
    ',
                            fontSize: '16px'
                        }}>
                            {item.name}
                        </span>
                        <span style={{
                            fontWeight: 'bold',
                            marginRight: '15px',
                            color: '#27ae60'
                        }}>
                            {item.price.toFixed(2)}DT
                        </span>
                        <button
                            onClick={() => deleteItem(item.id)}
                            style={{
                                padding: '6px 12px',
                                backgroundColor: '#e74c3c',
                                color: 'white',
                                border: 'none',
                                borderRadius: '3px',
                                cursor: 'pointer'
                            }}
                        >
                            Supprimer
                        </button>
                    </div>
                ))}
            </div>

            {/* Statistiques */}
            <div style={{
                marginTop: '20px',
                padding: '15px',
                backgroundColor: '#3498db',
                color: 'white',
                borderRadius: '5px',
                display: 'flex',
                justifyContent: 'space-between'
            }}>
                <div>
                    <strong>Total :</strong> {items.length} articles
    |
                    <strong> Achetés :</strong> {boughtCount}
                </div>
                <div>
                    <strong>Prix total :</strong> {totalPrice.toFixed
    (2)}DT
                </div>
            </div>
        </div>
    );
```

```
}

export default ShoppingListApp;
```

Listing 3 – src/ShoppingListApp.jsx

Concepts utilisés dans ce projet :

1. **useState multiple :** Pour gérer plusieurs états indépendants
2. **useEffect avec cleanup :** Pour le timer (important !)
3. **useEffect avec dépendances :** Pour localStorage
4. **Calculs dérivés :** totalPrice et boughtCount calculés à la volée
5. **Inputs contrôlés :** text et number

3.3 Projet 3 : Blog Interactif avec Tri et Recherche (60 min)

Objectif : Reprendre le blog de la semaine précédente et le rendre complètement interactif.

Nouvelles fonctionnalités :

- Ajouter des articles via formulaire
- Liker / Disliker
- Recherche par titre et contenu
- Tri par likes ou date
- Filtrer par auteur
- Supprimer des articles
- Persistance complète

3.3.1 Code Complet de l'Application

```
import { useState, useEffect } from 'react';

function BlogApp() {
    // Etats principaux
    const [articles, setArticles] = useState(() => {
        const saved = localStorage.getItem('blog-articles');
        if (saved) return JSON.parse(saved);
        return [
            {
                id: 1,
                title: "Introduction à React",
                author: "Alice",
                content: "React est une bibliothèque JavaScript...",
                likes: 5,
                createdAt: new Date().toISOString()
            },
            {
                id: 2,
                title: "Les Hooks en détail",
                author: "Bob",
                content: "Les hooks révolutionnent React...",
                likes: 3,
                createdAt: new Date().toISOString()
            }
        ];
    });

    // Etats de filtrage et tri
    const [searchTerm, setSearchTerm] = useState('');
    const [filterAuthor, setFilterAuthor] = useState('');
    const [sortBy, setSortBy] = useState('date'); // 'date' ou 'likes
    '

    // Etats du formulaire
```

```
    const [showForm, setShowForm] = useState(false);
    const [title, setTitle] = useState('');
    const [author, setAuthor] = useState('');
    const [content, setContent] = useState('');

    // Sauvegarder dans localStorage
    useEffect(() => {
        localStorage.setItem('blog-articles', JSON.stringify(articles
    ));
    }, [articles]);

    // Fonctions de gestion
    const addArticle = () => {
        if (!title.trim() || !author.trim() || !content.trim()) {
            alert('Tous les champs sont obligatoires');
            return;
        }

        const newArticle = {
            id: Date.now(),
            title,
            author,
            content,
            likes: 0,
            createdAt: new Date().toISOString()
        };

        setArticles([newArticle, ...articles]);
        setTitle('');
        setAuthor('');
        setContent('');
        setShowForm(false);
    };

    const handleLike = (id) => {
        setArticles(articles.map(article =>
            article.id === id
                ? {...article, likes: article.likes + 1}
                : article
        ));
    };

    const handleDelete = (id) => {
        if (window.confirm('Supprimer cet article ?')) {
            setArticles(articles.filter(article => article.id !== id)
    );
        }
    };

    // Filtrage et tri
    let filteredArticles = articles;
```

```
83
84      // Recherche
85      if (searchTerm) {
86          filteredArticles = filteredArticles.filter(article =>
87              article.title.toLowerCase().includes(searchTerm.
    toLowerCase()) ||
88              article.content.toLowerCase().includes(searchTerm.
    toLowerCase())
89          );
90      }
91
92      // Filtre par auteur
93      if (filterAuthor) {
94          filteredArticles = filteredArticles.filter(article =>
95              article.author === filterAuthor
96          );
97      }
98
99      // Tri
100     filteredArticles = [...filteredArticles].sort((a, b) => {
101         if (sortBy === 'likes') {
102             return b.likes - a.likes;
103         }
104         return new Date(b.createdAt) - new Date(a.createdAt);
105     });
106
107     // Liste unique des auteurs
108     const authors = [...new Set(articles.map(a => a.author))];
109
110     return (
111         <div style={{
112             maxWidth: '900px',
113             margin: '0 auto',
114             padding: '20px'
115         }}>
116             <h1 style={{ textAlign: 'center', color: '#2c3e50' }}>
117                 Mon Blog React Interactif
118             </h1>
119
120             {/* Barre d'actions */}
121             <div style={{
122                 backgroundColor: '#ecf0f1',
123                 padding: '20px',
124                 borderRadius: '10px',
125                 marginBottom: '20px'
126             }}>
127                 <div style={{
128                     display: 'grid',
129                     gridTemplateColumns: '1fr 1fr 1fr auto',
130                     gap: '10px',
131                     marginBottom: '15px'
```

```
                }}>
                    {/* Recherche */}
                    <input
                        type="text"
                        value={searchTerm}
                        onChange={(e) => setSearchTerm(e.target.value
)}
                        placeholder="Rechercher..."
                        style={{
                            padding: '10px',
                            border: '1px solid #bdc3c7',
                            borderRadius: '5px',
                            fontSize: '14px'
                        }}
                    />

                    {/* Filtre auteur */}
                    <select
                        value={filterAuthor}
                        onChange={(e) => setFilterAuthor(e.target.
value)}
                        style={{
                            padding: '10px',
                            border: '1px solid #bdc3c7',
                            borderRadius: '5px',
                            fontSize: '14px'
                        }}
                    >
                        <option value="">Tous les auteurs</option>
                        {authors.map(author => (
                            <option key={author} value={author}>
                                {author}
                            </option>
                        ))}
                    </select>

                    {/* Tri */}
                    <select
                        value={sortBy}
                        onChange={(e) => setSortBy(e.target.value)}
                        style={{
                            padding: '10px',
                            border: '1px solid #bdc3c7',
                            borderRadius: '5px',
                            fontSize: '14px'
                        }}
                    >
                        <option value="date">Plus récents</option>
                        <option value="likes">Plus likés</option>
                    </select>

```

```
                    {/* Bouton ajouter */}
                    <button
                        onClick={() => setShowForm(!showForm)}
                        style={{
                            padding: '10px 20px',
                            backgroundColor: '#27ae60',
                            color: 'white',
                            border: 'none',
                            borderRadius: '5px',
                            cursor: 'pointer',
                            fontWeight: 'bold'
                        }}
                    >
                        {showForm ? 'Annuler' : '+ Nouvel Article'}
                    </button>
                </div>

                <div style={{ fontSize: '14px', color: '#7f8c8d' }}>
                    {filteredArticles.length} article(s) affiché(s) sur {articles.length}
                </div>
            </div>

            {/* Formulaire d'ajout */}
            {showForm && (
                <div style={{
                    backgroundColor: '#fff',
                    padding: '20px',
                    borderRadius: '10px',
                    marginBottom: '20px',
                    border: '2px solid #27ae60'
                }}>
                    <h3>Nouvel Article</h3>
                    <input
                        type="text"
                        value={title}
                        onChange={(e) => setTitle(e.target.value)}
                        placeholder="Titre..."
                        style={{
                            width: '100%',
                            padding: '10px',
                            marginBottom: '10px',
                            border: '1px solid #ddd',
                            borderRadius: '5px',
                            fontSize: '16px'
                        }}
                    />
                    <input
                        type="text"
                        value={author}
                        onChange={(e) => setAuthor(e.target.value)}
```

```
                        placeholder="Auteur..."
                        style={{
                            width: '100%',
                            padding: '10px',
                            marginBottom: '10px',
                            border: '1px solid #ddd',
                            borderRadius: '5px',
                            fontSize: '16px'
                        }}
                    />
                    <textarea
                        value={content}
                        onChange={(e) => setContent(e.target.value)}
                        placeholder="Contenu..."
                        rows="5"
                        style={{
                            width: '100%',
                            padding: '10px',
                            marginBottom: '10px',
                            border: '1px solid #ddd',
                            borderRadius: '5px',
                            fontSize: '14px',
                            fontFamily: 'inherit'
                        }}
                    />
                    <button
                        onClick={addArticle}
                        style={{
                            padding: '12px 24px',
                            backgroundColor: '#27ae60',
                            color: 'white',
                            border: 'none',
                            borderRadius: '5px',
                            cursor: 'pointer',
                            fontSize: '16px',
                            fontWeight: 'bold'
                        }}
                    >
                        Publier
                    </button>
                </div>
            )}

            {/* Liste des articles */}
            <div>
                {filteredArticles.length === 0 ? (
                    <p style={{ textAlign: 'center', color: '#95a5a6'
, padding: '40px' }}>
                        Aucun article trouvé
                    </p>
                ) : (
```

```
                    filteredArticles.map(article => (
                        <article
                            key={article.id}
                            style={{
                                backgroundColor: 'white',
                                padding: '20px',
                                marginBottom: '20px',
                                borderRadius: '10px',
                                border: '1px solid #ddd'
                            }}
                        >
                            <h2 style={{ color: '#2c3e50', marginTop: 0 }}>
                                {article.title}
                            </h2>
                            <p style={{
                                color: '#7f8c8d',
                                fontSize: '14px',
                                marginBottom: '15px'
                            }}>
                                Par {article.author} | {
                                    new Date(article.createdAt).toLocaleDateString('fr-FR')
                                }
                            </p>
                            <p style={{ lineHeight: '1.6', color: '#34495e' }}>
                                {article.content}
                            </p>
                            <div style={{
                                display: 'flex',
                                alignItems: 'center',
                                gap: '10px',
                                marginTop: '15px',
                                paddingTop: '15px',
                                borderTop: '1px solid #ecf0f1'
                            }}>
                                <button
                                    onClick={() => handleLike(article.id)}
                                    style={{
                                        padding: '8px 16px',
                                        backgroundColor: '#3498db',
                                        color: 'white',
                                        border: 'none',
                                        borderRadius: '5px',
                                        cursor: 'pointer',
                                        display: 'flex',
                                        alignItems: 'center',
                                        gap: '5px'
                                    }}
```

```
                                    >
                                        {article.likes}
                                    </button>
                                    <button
                                        onClick={() => handleDelete(
    article.id)}
                                        style={{
                                            padding: '8px 16px',
                                            backgroundColor: '#e74c3c',
                                            color: 'white',
                                            border: 'none',
                                            borderRadius: '5px',
                                            cursor: 'pointer',
                                            marginLeft: 'auto'
                                        }}
                                    >
                                        Supprimer
                                    </button>
                                </div>
                            </article>
                        ))
                    )}
                </div>
            </div>
        );
}

export default BlogApp;
```

Listing 4 – src/BlogApp.jsx - Version complète

Récapitulatif des techniques utilisées :

- **useState avec initialisation lazy :** Charge depuis localStorage au montage
- **useEffect pour persistance :** Sauvegarde automatique
- **Filtrage multiple :** Recherche + auteur + tri
- **Immutabilité stricte :** map(), filter(), spread operator
- **Affichage conditionnel :** Formulaire et message vide
- **Calculs dérivés :** Liste d'auteurs unique avec Set

4 Récapitulatif et Concepts Clés

Les 10 Règles d'Or des Hooks

1. **Toujours au top level :** Jamais dans des conditions, boucles ou fonctions
2. **Immutabilité stricte :** Toujours créer de nouveaux objets/tableaux
3. **Forme fonctionnelle :** `setState(prev => ...)` pour updates multiples
4. **Tableau de dépendances :** Inclure toutes les valeurs utilisées
5. **Cleanup obligatoire :** Pour timers, événements, abonnements
6. **Initialisation lazy :** `useState(() => ...)` pour calculs coûteux
7. **useReducer pour complexité :** Quand plusieurs états liés
8. **localStorage :** useEffect pour persistance
9. **Inputs contrôlés :** value + onChange toujours liés
10. **Clés stables :** Utiliser des IDs uniques, pas les index

Patterns à retenir :

```
// Ajouter à un tableau
setItems([...items, newItem]);

// Supprimer d'un tableau
setItems(items.filter(item => item.id !== id));

// Modifier dans un tableau
setItems(items.map(item =>
    item.id === id ? {...item, prop: value} : item
));

// Modifier un objet
setUser({...user, name: "Bob"});

// Timer avec cleanup
useEffect(() => {
    const timer = setInterval(() => {...}, 1000);
    return () => clearInterval(timer);
}, [deps]);

// localStorage
useEffect(() => {
    localStorage.setItem('key', JSON.stringify(data));
}, [data]);
```

5 Travail à Rendre

Projet Final

Créer une application complète "Gestionnaire de Projets" combinant les trois hooks.

Fonctionnalités obligatoires :

1. **useReducer : Gérer les projets (ajouter, modifier statut, supprimer)**
2. **useEffect : Persistance localStorage + timer par projet**
3. **useState : Formulaires, filtres, recherche**
4. **Chaque projet a : titre, description, statut (todo/doing/-done), deadline**
5. **Filtrer par statut**
6. **Trier par deadline**
7. **Recherche dans titre/description**
8. **Timer Pomodoro par projet**
9. **Statistiques visuelles (nombre par statut)**

Livrables :

- **Code source complet**
- **Captures d'écran de toutes les fonctionnalités**
- **Explication du reducer (tous les cas)**
- **Explication de chaque useEffect**
- **README avec instructions**

Échéance : La veille de la prochaine séance à 23h59

Ce travail compte pour 20% de la note finale.

6 Ressources

6.1 Documentation Officielle

- React Documentation : `https://react.dev`
- useState : `https://react.dev/reference/react/useState`
- useEffect : `https://react.dev/reference/react/useEffect`
- useReducer : `https://react.dev/reference/react/useReducer`
- Rules of Hooks : `https://react.dev/reference/rules/rules-of-hooks`

6.2 Outils

— ESLint Plugin React Hooks (détection des erreurs)
— React DevTools (inspection des états)