

Cours MERN - Semaine 7

Maîtriser les Hooks Essentiels React

`useState`, `useEffect` et `useReducer`

Abdelweheb GUEDDDES & 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 : <code>useState</code> - Ajouter de la mémoire	2
2.1.1 L'Immutabilité : La Règle d'Or	2
2.2 Hook 2 : <code>useEffect</code> - Les effets de bord	3
2.3 Hook 3 : <code>useReducer</code> - État complexe	3
3 Partie 2 : Projets Pratiques (2h30)	5
3.1 Projet 1 : Gestionnaire de Tâches avec <code>useReducer</code> (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 :

```
1 const [state, setState] = useState(initialValue);
2
3 // Exemples
4 const [count, setCount] = useState(0);
5 const [name, setName] = useState('');
6 const [items, setItems] = useState([]);
7 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 :

```
1 // Objets
2 user.name = "Bob";
3 setUser(user); // React ne détecte pas le changement
4
5 // Tableaux
6 items.push(newItem);
7 setItems(items); // React ne détecte pas le changement
```

CORRECT - Immutabilité :

```
1 // Objets
2 setUser({ ...user, name: "Bob" }); // Nouvel objet
3
4 // Tableaux
5 setItems([...items, newItem]); // Nouveau tableau
```

Opérations immutables sur les tableaux :

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

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 :

```
1 useEffect(() => {  
2   // Code à exécuter  
3  
4   return () => {  
5     // Nettoyage (optionnel)  
6   };  
7 }, [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 :

```
1 const [state, dispatch] = useReducer(reducer, initialState);  
2  
3 // Fonction reducer  
4 function reducer(state, action) {  
5   switch (action.type) {  
6     case 'ACTION_NAME':  
7       return {...state, /* modifications */};  
5
```

```
8         default:
9             return state;
10     }
11 }
12
13 // Utilisation
14 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

```
1 export const initialState = {
2   todos: [],
3   filter: 'all' // 'all', 'active', 'completed'
4 };
5
6 export function todoReducer(state, action) {
7   switch (action.type) {
8     case 'ADD_TODO':
9       return {
10         ...state,
11         todos: [
12           {
13             id: Date.now(),
14             text: action.payload,
15             completed: false,
16             createdAt: new Date().toISOString()
17           },
18           ...state.todos
19         ]
20       };
21
22     case 'TOGGLE_TODO':
23       return {
24         ...state,
25         todos: state.todos.map(todo =>
26           todo.id === action.payload
27             ? {...todo, completed: !todo.completed}
28             : todo
29         )
30       };
31
32     case 'DELETE_TODO':
33       return {
34         ...state,
35         todos: state.todos.filter(todo =>
```

```
36         todo.id !== action.payload
37     )
38 };
39
40 case 'SET_FILTER':
41     return {
42         ...state,
43         filter: action.payload
44     };
45
46 case 'LOAD_TODOS':
47     return {
48         ...state,
49         todos: action.payload
50     };
51
52 default:
53     return state;
54 }
55 }
```

Listing 1 – src/reducers/todoReducer.js

3.1.2 Étape 2 : Le Composant Principal

```
1 import { useReducer, useState, useEffect } from 'react';
2 import { todoReducer, initialState } from '../reducers/todoReducer';
3
4 function App() {
5     const [state, dispatch] = useReducer(todoReducer, initialState);
6     const [input, setInput] = useState('');
7
8     // Charger depuis localStorage au montage
9     useEffect(() => {
10         const saved = localStorage.getItem('todos');
11         if (saved) {
12             dispatch({
13                 type: 'LOAD_TODOS',
14                 payload: JSON.parse(saved)
15             });
16         }
17     }, []);
18
19     // Sauvegarder dans localStorage à chaque modification
20     useEffect(() => {
21         if (state.todos.length > 0) {
22             localStorage.setItem('todos', JSON.stringify(state.todos))
23         }
24     }, [state.todos]);
25 }
```

```
26   const addTodo = () => {
27     if (input.trim()) {
28       dispatch({ type: 'ADD_TODO', payload: input });
29       setInput('');
30     }
31   };
32
33   // Filtrer les todos
34   const filteredTodos = state.todos.filter(todo => {
35     if (state.filter === 'active') return !todo.completed;
36     if (state.filter === 'completed') return todo.completed;
37     return true;
38   });
39
40   return (
41     <div style={{
42       maxWidth: '600px',
43       margin: '50px auto',
44       padding: '20px',
45       fontFamily: 'Arial, sans-serif'
46     }}>
47       <h1 style={{ textAlign: 'center', color: '#2c3e50' }}>
48         Ma Todo List
49       </h1>
50
51       { /* Formulaire d'ajout */ }
52       <div style={{
53         display: 'flex',
54         gap: '10px',
55         marginBottom: '20px'
56       }}>
57         <input
58           type="text"
59           value={input}
60           onChange={(e) => setInput(e.target.value)}
61           onKeyDown={(e) => e.key === 'Enter' && addTodo()}
62
63           placeholder="Nouvelle tâche..."
64           style={{
65             flex: 1,
66             padding: '12px',
67             fontSize: '16px',
68             border: '2px solid #ddd',
69             borderRadius: '5px'
70           }}
71         />
72         <button
73           onClick={addTodo}
74           style={{
75             padding: '12px 24px',
76             backgroundColor: '#27ae60',
```

```
76         color: 'white',
77         border: 'none',
78         borderRadius: '5px',
79         cursor: 'pointer',
80         fontSize: '16px',
81         fontWeight: 'bold'
82     }}
83     >
84     Ajouter
85 </button>
86 </div>
87
88 { /* Filtres */ }
89 <div style={{
90     display: 'flex',
91     gap: '10px',
92     marginBottom: '20px',
93     justifyContent: 'center'
94 }}>
95     {[ 'all', 'active', 'completed' ].map( filter => (
96         <button
97             key={filter}
98             onClick={() => dispatch({
99                 type: 'SET_FILTER',
100                 payload: filter
101             })}
102             style={{
103                 padding: '8px 16px',
104                 backgroundColor: state.filter === filter
105                     ? '#3498db'
106                     : '#ecf0f1',
107                 color: state.filter === filter
108                     ? 'white'
109                     : '#2c3e50',
110                 border: 'none',
111                 borderRadius: '5px',
112                 cursor: 'pointer',
113                 fontWeight: state.filter === filter
114                     ? 'bold'
115                     : 'normal'
116             }}
117         >
118             { filter === 'all' ? 'Toutes' :
119             filter === 'active' ? 'Actives' : 'Complété
120 es' }
121         </button>
122     ) ) }
123 </div>
124
125 { /* Liste des tâches */ }
126 <div>
```



```
126     {filteredTodos.length === 0 ? (  
127       <p style={{  
128         textAlign: 'center',  
129         color: '#95a5a6',  
130         padding: '40px'  
131       }}>  
132         Aucune tâche à afficher  
133       </p>  
134     ) : (  
135       filteredTodos.map(todo => (  
136         <div  
137           key={todo.id}  
138           style={{  
139             display: 'flex',  
140             alignItems: 'center',  
141             padding: '15px',  
142             marginBottom: '10px',  
143             backgroundColor: todo.completed  
144               ? '#d5f4e6'  
145               : 'white',  
146             border: '1px solid #ddd',  
147             borderRadius: '5px'  
148           }}  
149         >  
150           <input  
151             type="checkbox"  
152             checked={todo.completed}  
153             onChange={() => dispatch({  
154               type: 'TOGGLE_TODO',  
155               payload: todo.id  
156             })}  
157             style={{  
158               marginRight: '15px',  
159               width: '20px',  
160               height: '20px',  
161               cursor: 'pointer'  
162             }}  
163           />  
164           <span style={{  
165             flex: 1,  
166             fontSize: '16px',  
167             textDecoration: todo.completed  
168               ? 'line-through'  
169               : 'none',  
170             color: todo.completed  
171               ? '#95a5a6'  
172               : '#2c3e50'  
173           }}>  
174             {todo.text}  
175           </span>  
176           <button
```

```

177         onClick={() => dispatch({
178             type: 'DELETE_TODO',
179             payload: todo.id
180         })}
181         style={{
182             padding: '6px 12px',
183             backgroundColor: '#e74c3c',
184             color: 'white',
185             border: 'none',
186             borderRadius: '3px',
187             cursor: 'pointer'
188         }}
189     >
190         Supprimer
191     </button>
192 </div>
193 ))
194 })
195 </div>
196
197 { /* Statistiques */ }
198 <div style={{
199     marginTop: '20px',
200     textAlign: 'center',
201     padding: '15px',
202     backgroundColor: '#ecf0f1',
203     borderRadius: '5px'
204 }}>
205     <strong>Total :</strong> {state.todos.length} |
206     <strong> Complétées :</strong> {
207         state.todos.filter(t => t.completed).length
208     }
209 </div>
210 </div>
211 );
212 }
213
214 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

```
1 import { useState, useEffect } from 'react';
2
3 function ShoppingListApp() {
4   // Etats pour la liste
5   const [items, setItems] = useState([]);
6   const [inputValue, setInputValue] = useState('');
7   const [inputPrice, setInputPrice] = useState('');
8
9   // Etats pour le timer Pomodoro
10  const [minutes, setMinutes] = useState(25);
11  const [seconds, setSeconds] = useState(0);
12  const [isActive, setIsActive] = useState(false);
13
14  // Charger la liste depuis localStorage
15  useEffect(() => {
16    const saved = localStorage.getItem('shopping-list');
17    if (saved) {
18      setItems(JSON.parse(saved));
19    }
20  }, []);
21
22  // Sauvegarder la liste
23  useEffect(() => {
24    localStorage.setItem('shopping-list', JSON.stringify(items));
25  }, [items]);
26
27  // Timer Pomodoro
28  useEffect(() => {
29    let interval = null;
30
31    if (isActive) {
32      interval = setInterval(() => {
33        if (seconds === 0) {
34          if (minutes === 0) {
35            setIsActive(false);
36            alert('Temps de course écoulé !');
37          } else {
38            setMinutes(minutes - 1);
39            setSeconds(59);
```

```
40         }
41       } else {
42         setSeconds(seconds - 1);
43       }
44     }, 1000);
45   }
46
47   // Nettoyage : arrêter l'intervalle
48   return () => {
49     if (interval) clearInterval(interval);
50   };
51 }, [isActive, minutes, seconds]);
52
53 const addItem = () => {
54   if (inputValue.trim()) {
55     setItems([
56       ...items,
57       {
58         id: Date.now(),
59         name: inputValue,
60         price: inputPrice ? parseFloat(inputPrice) : 0,
61         bought: false
62       }
63     ]);
64     setInputValue('');
65     setInputPrice('');
66   }
67 };
68
69 const toggleBought = (id) => {
70   setItems(items.map(item =>
71     item.id === id
72       ? {...item, bought: !item.bought}
73       : item
74   ));
75 };
76
77 const deleteItem = (id) => {
78   setItems(items.filter(item => item.id !== id));
79 };
80
81 const resetTimer = () => {
82   setIsActive(false);
83   setMinutes(25);
84   setSeconds(0);
85 };
86
87 const totalPrice = items.reduce((sum, item) => sum + item.price,
88 0);
89 const boughtCount = items.filter(item => item.bought).length;
```

```

90     return (
91       <div style={{
92         maxWidth: '700px',
93         margin: '50px auto',
94         padding: '20px'
95       }}>
96         <h1 style={{ textAlign: 'center' }}>
97           Liste de Courses
98         </h1>
99
100         { /* Timer Pomodoro */ }
101         <div style={{
102           backgroundColor: '#34495e',
103           color: 'white',
104           padding: '20px',
105           borderRadius: '10px',
106           marginBottom: '30px',
107           textAlign: 'center'
108         }}>
109           <h3>Timer de Course</h3>
110           <div style={{
111             fontSize: '48px',
112             fontWeight: 'bold',
113             margin: '20px 0',
114             color: isActive ? '#e74c3c' : 'white'
115           }}>
116             {String(minutes).padStart(2, '0')}:
117             {String(seconds).padStart(2, '0')}
118           </div>
119           <div style={{ display: 'flex', gap: '10px',
justifyContent: 'center' }}>
120             <button
121               onClick={() => setIsActive(!isActive)}
122               style={{
123                 padding: '10px 20px',
124                 backgroundColor: isActive ? '#f39c12' : '
#27ae60',
125                 color: 'white',
126                 border: 'none',
127                 borderRadius: '5px',
128                 cursor: 'pointer',
129                 fontSize: '16px'
130               }}
131             >
132               {isActive ? 'Pause' : 'Démarrer'}
133             </button>
134             <button
135               onClick={resetTimer}
136               style={{
137                 padding: '10px 20px',
138                 backgroundColor: '#95a5a6',

```

```
139         color: 'white',
140         border: 'none',
141         borderRadius: '5px',
142         cursor: 'pointer',
143         fontSize: '16px'
144     }}
145     >
146     Reset
147 </button>
148 </div>
149 </div>
150
151 { /* Formulaire d'ajout */ }
152 <div style={{
153     backgroundColor: '#ecf0f1',
154     padding: '20px',
155     borderRadius: '10px',
156     marginBottom: '20px'
157 }}>
158     <div style={{ display: 'flex', gap: '10px',
marginBottom: '10px' }}>
159         <input
160             type="text"
161             value={inputValue}
162             onChange={(e) => setInputValue(e.target.value
163             ))
164             onKeyDown={(e) => e.key === 'Enter' &&
addItem()}
165             placeholder="Nom de l'article..."
166             style={{
167                 flex: 2,
168                 padding: '10px',
169                 fontSize: '16px',
170                 border: '1px solid #bdc3c7',
171                 borderRadius: '5px'
172             }}
173         />
174         <input
175             type="number"
176             value={inputPrice}
177             onChange={(e) => setInputPrice(e.target.value
178             ))
179             onKeyDown={(e) => e.key === 'Enter' &&
addItem()}
180             placeholder="Prix..."
181             step="0.01"
182             style={{
183                 flex: 1,
184                 padding: '10px',
185                 fontSize: '16px',
186                 border: '1px solid #bdc3c7',
```

```
185         borderRadius: '5px'
186     }}
187     />
188     <button
189         onClick={addItem}
190         style={{
191             padding: '10px 20px',
192             backgroundColor: '#3498db',
193             color: 'white',
194             border: 'none',
195             borderRadius: '5px',
196             cursor: 'pointer',
197             fontSize: '16px'
198         }}
199     >
200         Ajouter
201     </button>
202 </div>
203 </div>
204
205 { /* Liste des articles */ }
206 <div>
207     {items.map(item => (
208         <div
209             key={item.id}
210             style={{
211                 display: 'flex',
212                 alignItems: 'center',
213                 padding: '15px',
214                 marginBottom: '10px',
215                 backgroundColor: item.bought ? '#d5f4e6'
216                 : 'white',
217                 border: '1px solid #ddd',
218                 borderRadius: '5px'
219             }}
220         >
221             <input
222                 type="checkbox"
223                 checked={item.bought}
224                 onChange={() => toggleBought(item.id)}
225                 style={{
226                     marginRight: '15px',
227                     width: '20px',
228                     height: '20px'
229                 }}
230             />
231             <span style={{
232                 flex: 1,
233                 textDecoration: item.bought ? 'line-through' : 'none',
234                 color: item.bought ? '#95a5a6' : '#2c3e50'
```

```
234         ,
235         fontSize: '16px'
236     }}>
237     {item.name}
238 </span>
239 <span style={{
240     fontWeight: 'bold',
241     marginRight: '15px',
242     color: '#27ae60'
243 }}>
244     {item.price.toFixed(2)}DT
245 </span>
246 <button
247     onClick={() => deleteItem(item.id)}
248     style={{
249         padding: '6px 12px',
250         backgroundColor: '#e74c3c',
251         color: 'white',
252         border: 'none',
253         borderRadius: '3px',
254         cursor: 'pointer'
255     }}
256 >
257     Supprimer
258 </button>
259 </div>
260 ))}
261 </div>
262 { /* Statistiques */ }
263 <div style={{
264     marginTop: '20px',
265     padding: '15px',
266     backgroundColor: '#3498db',
267     color: 'white',
268     borderRadius: '5px',
269     display: 'flex',
270     justifyContent: 'space-between'
271 }}>
272     <div>
273         <strong>Total :</strong> {items.length} articles
274         |
275         <strong> Achetés :</strong> {boughtCount}
276     </div>
277     <div>
278         <strong>Prix total :</strong> {totalPrice.toFixed
279         (2)}DT
280     </div>
281 </div>
282 );
```



```
282 }  
283  
284 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

```
1 import { useState, useEffect } from 'react';
2
3 function BlogApp() {
4   // Etats principaux
5   const [articles, setArticles] = useState(() => {
6     const saved = localStorage.getItem('blog-articles');
7     if (saved) return JSON.parse(saved);
8     return [
9       {
10         id: 1,
11         title: "Introduction à React",
12         author: "Alice",
13         content: "React est une bibliothèque JavaScript...",
14         likes: 5,
15         createdAt: new Date().toISOString()
16       },
17       {
18         id: 2,
19         title: "Les Hooks en détail",
20         author: "Bob",
21         content: "Les hooks révolutionnent React...",
22         likes: 3,
23         createdAt: new Date().toISOString()
24       }
25     ];
26   });
27
28   // Etats de filtrage et tri
29   const [searchTerm, setSearchTerm] = useState('');
30   const [filterAuthor, setFilterAuthor] = useState('');
31   const [sortBy, setSortBy] = useState('date'); // 'date' ou 'likes'
32
33   // Etats du formulaire
```

```
34  const [showForm, setShowForm] = useState(false);
35  const [title, setTitle] = useState('');
36  const [author, setAuthor] = useState('');
37  const [content, setContent] = useState('');
38
39  // Sauvegarder dans localStorage
40  useEffect(() => {
41    localStorage.setItem('blog-articles', JSON.stringify(articles
42  ));
43  }, [articles]);
44
45  // Fonctions de gestion
46  const addArticle = () => {
47    if (!title.trim() || !author.trim() || !content.trim()) {
48      alert('Tous les champs sont obligatoires');
49      return;
50    }
51
52    const newArticle = {
53      id: Date.now(),
54      title,
55      author,
56      content,
57      likes: 0,
58      createdAt: new Date().toISOString()
59    };
60
61    setArticles([newArticle, ...articles]);
62    setTitle('');
63    setAuthor('');
64    setContent('');
65    setShowForm(false);
66  };
67
68  const handleLike = (id) => {
69    setArticles(articles.map(article =>
70      article.id === id
71        ? {...article, likes: article.likes + 1}
72        : article
73    ));
74  };
75
76  const handleDelete = (id) => {
77    if (window.confirm('Supprimer cet article ?')) {
78      setArticles(articles.filter(article => article.id !== id)
79    );
80    }
81  };
82
83  // Filtrage et tri
84  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'
```

```
132     }}>
133     { /* Recherche */ }
134     <input
135       type="text"
136       value={searchTerm}
137       onChange={(e) => setSearchTerm(e.target.value
138     )}
139
140     placeholder="Rechercher..."
141     style={{
142       padding: '10px',
143       border: '1px solid #bdc3c7',
144       borderRadius: '5px',
145       fontSize: '14px'
146     }}
147   />
148
149   { /* Filtre auteur */ }
150   <select
151     value={filterAuthor}
152     onChange={(e) => setFilterAuthor(e.target.
153   value)}
154
155     style={{
156       padding: '10px',
157       border: '1px solid #bdc3c7',
158       borderRadius: '5px',
159       fontSize: '14px'
160     }}
161   >
162     <option value="">Tous les auteurs</option>
163     { authors.map(author => (
164       <option key={author} value={author}>
165         {author}
166       </option>
167     ))}
168   </select>
169
170   { /* Tri */ }
171   <select
172     value={sortBy}
173     onChange={(e) => setSortBy(e.target.value)}
174     style={{
175       padding: '10px',
176       border: '1px solid #bdc3c7',
177       borderRadius: '5px',
178       fontSize: '14px'
179     }}
180   >
181     <option value="date">Plus récents</option>
182     <option value="likes">Plus likés</option>
183   </select>
```

```
181      {/* Bouton ajouter */}
182      <button
183        onClick={() => setShowForm(!showForm)}
184        style={{
185          padding: '10px 20px',
186          backgroundColor: '#27ae60',
187          color: 'white',
188          border: 'none',
189          borderRadius: '5px',
190          cursor: 'pointer',
191          fontWeight: 'bold'
192        }}
193      >
194        {showForm ? 'Annuler' : '+ Nouvel Article'}
195      </button>
196    </div>
197
198    <div style={{ fontSize: '14px', color: '#7f8c8d' }}>
199      {filteredArticles.length} article(s) affiché(s)
200    </div>
201  </div>
202
203  {/* Formulaire d'ajout */}
204  {showForm && (
205    <div style={{
206      backgroundColor: '#fff',
207      padding: '20px',
208      borderRadius: '10px',
209      marginBottom: '20px',
210      border: '2px solid #27ae60'
211    }}>
212    <h3>Nouvel Article</h3>
213    <input
214      type="text"
215      value={title}
216      onChange={(e) => setTitle(e.target.value)}
217      placeholder="Titre..."
218      style={{
219        width: '100%',
220        padding: '10px',
221        marginBottom: '10px',
222        border: '1px solid #ddd',
223        borderRadius: '5px',
224        fontSize: '16px'
225      }}
226    />
227    <input
228      type="text"
229      value={author}
230      onChange={(e) => setAuthor(e.target.value)}

```

```
231         placeholder="Auteur..."
232         style={{
233             width: '100%',
234             padding: '10px',
235             marginBottom: '10px',
236             border: '1px solid #ddd',
237             borderRadius: '5px',
238             fontSize: '16px'
239         }}
240     />
241     <textarea
242         value={content}
243         onChange={(e) => setContent(e.target.value)}
244         placeholder="Contenu..."
245         rows="5"
246         style={{
247             width: '100%',
248             padding: '10px',
249             marginBottom: '10px',
250             border: '1px solid #ddd',
251             borderRadius: '5px',
252             fontSize: '14px',
253             fontFamily: 'inherit'
254         }}
255     />
256     <button
257         onClick={addArticle}
258         style={{
259             padding: '12px 24px',
260             backgroundColor: '#27ae60',
261             color: 'white',
262             border: 'none',
263             borderRadius: '5px',
264             cursor: 'pointer',
265             fontSize: '16px',
266             fontWeight: 'bold'
267         }}
268     >
269         Publier
270     </button>
271 </div>
272 )}
273
274 { /* Liste des articles */ }
275 <div>
276     {filteredArticles.length === 0 ? (
277         <p style={{ textAlign: 'center', color: '#95a5a6'
278 , padding: '40px' }}>
279         Aucun article trouvé
280         </p>
281     ) : (
```

```

281         filteredArticles.map( article => (
282             <article
283                 key={article.id}
284                 style={{
285                     backgroundColor: 'white',
286                     padding: '20px',
287                     marginBottom: '20px',
288                     borderRadius: '10px',
289                     border: '1px solid #ddd'
290                 }}
291             >
292                 <h2 style={{ color: '#2c3e50', marginTop:
0 }}>
293                     { article.title }
294                 </h2>
295                 <p style={{
296                     color: '#7f8c8d',
297                     fontSize: '14px',
298                     marginBottom: '15px'
299                 }}>
300                     Par { article.author } | {
301                         new Date( article.createdAt ).
toLocaleDateString( 'fr-FR' )
302                     }
303                 </p>
304                 <p style={{ lineHeight: '1.6', color: '#34495e' }}>
305                     { article.content }
306                 </p>
307                 <div style={{
308                     display: 'flex',
309                     alignItems: 'center',
310                     gap: '10px',
311                     marginTop: '15px',
312                     padding: '15px',
313                     borderTop: '1px solid #ecf0f1'
314                 }}>
315                     <button
316                         onClick={() => handleLike( article
.id )}
317                         style={{
318                             padding: '8px 16px',
319                             backgroundColor: '#3498db',
320                             color: 'white',
321                             border: 'none',
322                             borderRadius: '5px',
323                             cursor: 'pointer',
324                             display: 'flex',
325                             alignItems: 'center',
326                             gap: '5px'
327                             }}

```



```
328 >
329         { article.likes }
330     </button>
331     <button
332         onClick={() => handleDelete(
333             article.id)}
334         style={{
335             padding: '8px 16px',
336             backgroundColor: '#e74c3c',
337             color: 'white',
338             border: 'none',
339             borderRadius: '5px',
340             cursor: 'pointer',
341             marginLeft: 'auto'
342         }}
343     > Supprimer
344     </button>
345 </div>
346 </article>
347 ))
348 })
349 </div>
350 </div>
351 );
352 }
353
354 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 :

```
1 // Ajouter à un tableau
2 setItems([...items, newItem]);
3
4 // Supprimer d'un tableau
5 setItems(items.filter(item => item.id !== id));
6
7 // Modifier dans un tableau
8 setItems(items.map(item =>
9   item.id === id ? {...item, prop: value} : item
10 ));
11
12 // Modifier un objet
13 setUser({...user, name: "Bob"});
14
15 // Timer avec cleanup
16 useEffect(() => {
17   const timer = setInterval(() => {...}, 1000);
18   return () => clearInterval(timer);
19 }, [deps]);
20
21 // localStorage
22 useEffect(() => {
23   localStorage.setItem('key', JSON.stringify(data));
24 }, [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** : Formulaire, 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)