CRUD React para 'articles' usando useReducer, acciones separadas y async/await

Generado: 2025-10-21 12:33:57

Referencia al fichero de rutas del backend subido por el usuario:

article.js (rutas Express) -> citation: ■filecite■turn0file0■

== File: src/utils/api/API.js ==

```
// 1. Obtener URL desde la variable de entorno
const BASE_URL = import.meta.env.VITE_API_BASE_URL;

// 2. Función Genérica de conexión a la API
export const API = async({ endpoint, method = "GET", body, hasBody = false }) => {
    // Para este caso no hacen falta los headers. Únicamente para cuando hay que enviar credenciales
.
    // let headers = {};
    // if(localStorage.getItem("hhToken")) {
    //     headers["Authorization"] = `Bearer ${localStorage.getItem("hhToken")}`;
    // }


    let requestBody = null;

    // Manejor del content-type
    if(hasBody) {
        headers["Content-Type"] = "application/json";
        requestBody = JSON.stringify(body);
    }

    // Construir la URL
    const url = BASE_URL + endpoint;

    try {
        const res = await fetch(url, {
            method,
            // headers,
            body: requestBody
        });
        // Comprobar error
        if(!res.ok) {
            return { error: response };
        }
        // Parsear resultado a json
        const response = await res.json();
        // Devolver resultado
        return { response };
    } catch (error) {
        return { error:{ message: error || "Error de conexión" } };
    }
};
```

== File: src/state/article.actions.js ==

```
// Archivo de acciones para el reducer de artículos
// Exportamos constantes que representan los tipos de acción
```

```
export const ACTIONS = {
    // Inicializa la carga de artículos
    FETCH_INIT: 'FETCH_INIT',
    // Cuando la carga fue exitosa
    FETCH_SUCCESS: 'FETCH_SUCCESS',
    // Cuando ocurre un error en la carga
    FETCH_FAILURE: 'FETCH_FAILURE',
    // Añadir un artículo nuevo
    ADD_ARTICLE: 'ADD_ARTICLE',
    // Actualizar un artículo existente
    UPDATE_ARTICLE: 'UPDATE_ARTICLE',
    // Eliminar un artículo por id
    DELETE_ARTICLE: 'DELETE_ARTICLE',
    // Seleccionar un artículo para edición
    SET_EDITING: 'SET_EDITING',
    // Limpiar el artículo en edición
    CLEAR_EDITING: 'CLEAR_EDITING'
};


== File: src/state/article.reducer.js ==

import { ACTIONS } from './article.actions.js';

// Reducer para manejar el estado de artículos
export const articleReducer = (state, action) => {
    // Switch para cada tipo de acción
    switch(action.type) {
        // Iniciamos la carga: ponemos loading true y limpiamos error
        case ACTIONS.FETCH_INIT:
            return { ...state, loading: true, error: null };
        // Carga exitosa: colocamos los datos en articles y apagamos loading
        case ACTIONS.FETCH_SUCCESS:
            return { ...state, loading: false, articles: action.payload, error: null };
        // Error en carga: almacenamos el error y apagamos loading
        case ACTIONS.FETCH_FAILURE:
            return { ...state, loading: false, error: action.payload };
        // Añadir artículo: lo añadimos al array articles
        case ACTIONS.ADD_ARTICLE:
            return { ...state, articles: [ action.payload, ...state.articles ] };
        // Actualizar artículo: reemplazamos por id
        case ACTIONS.UPDATE_ARTICLE:
            return {
                ...state,
                articles: state.articles.map(a => a._id === action.payload._id ? action.payload : a)
,
                editing: null
            };
        // Eliminar artículo: filtramos por id
        case ACTIONS.DELETE_ARTICLE:
            return { ...state, articles: state.articles.filter(a => a._id !== action.payload) };
        // Seleccionar artículo para edición
        case ACTIONS.SET_EDITING:
            return { ...state, editing: action.payload };
        // Limpiar edición
        case ACTIONS.CLEAR_EDITING:
            return { ...state, editing: null };
        // Default: devolvemos el estado actual
        default:
            return state;
```

```
    }
};


== File: src/services/articleService.js ==

// Servicio que usa la función API proporcionada para interactuar con el backend
import { API } from '../utils/api/API.js';

// Obtener todos los artículos
export const fetchArticles = async () => {
    // Llamada a API genérica usando async/await
    const { response, error } = await API({ endpoint: '/articles', method: 'GET' });
    // Devolvemos los datos o lanzamos error para manejarlo en el componente
    if(error) throw error;
    return response;
};

// Obtener artículo por id
export const fetchArticleById = async (id) => {
    const { response, error } = await API({ endpoint: `/articles/${id}`, method: 'GET' });
    if(error) throw error;
    return response;
};

// Crear nuevo artículo (hasBody true)
export const createArticle = async (body) => {
    const { response, error } = await API({ endpoint: '/articles', method: 'POST', body, hasBody: tr
ue });
    if(error) throw error;
    return response;
};

// Actualizar artículo por id
export const updateArticleById = async (id, body) => {
    const { response, error } = await API({ endpoint: `/articles/${id}`, method: 'PUT', body, hasBod
y: true });
    if(error) throw error;
    return response;
};

// Eliminar artículo por id
export const deleteArticleById = async (id) => {
    const { response, error } = await API({ endpoint: `/articles/${id}`, method: 'DELETE' });
    if(error) throw error;
    return response;
};


== File: src/components/ArticleList/ArticleList.jsx ==

import React, { useEffect } from 'react'; // Import React y useEffect desde React
import './ArticleList.css'; // Importar estilos del componente

// Componente funcional para listar artículos
const ArticleList = ({ state, dispatch, actions, services }) => {
    // useEffect para cargar los artículos al montar el componente
    useEffect(() => {
        // Definimos una función async dentro del efecto
        const load = async () => {
```

```
            // Disparamos la acción de inicio de fetch
            dispatch({ type: actions.FETCH_INIT });
            try {
                // Llamada al servicio para obtener artículos (async/await)
                const data = await services.fetchArticles();
                // Dispatch success con los datos
                dispatch({ type: actions.FETCH_SUCCESS, payload: data });
            } catch (err) {
                // Dispatch failure con el error
                dispatch({ type: actions.FETCH_FAILURE, payload: err });
            }
        };
        // Ejecutar carga
        load();
    }, [dispatch, actions, services]); // Dependencias del useEffect

    // Renderizado del componente
    return (
        <div className='article-list'>
            {/* Si loading true mostramos mensaje */}
            {state.loading && <p>Cargando artículos...</p>}
            {/* Si hay error mostramos mensaje */}
            {state.error && <p className='error'>Error: {state.error.message || JSON.stringify(state
.error)}</p>}
            {/* Listado de artículos */}
            <ul>
                {state.articles.map(article => (
                    <li key={article._id}>
                        {/* Usamos ArticleItem desde props.services para simplificar */}
                        <div className='article-item'>
                            <h3>{article.title}</h3>
                            <p>{article.content}</p>
                            <div className='actions'>
                                <button onClick={() => dispatch({ type: actions.SET_EDITING, payload
: article })}>Editar</button>
                                <button onClick={async () => {
                                    // Confirmación y eliminación usando servicio
                                    if(!confirm('¿Eliminar artículo?')) return;
                                    try {
                                        await services.deleteArticleById(article._id);
                                        dispatch({ type: actions.DELETE_ARTICLE, payload: article._i
d });

                                    } catch(e) {
                                        alert('Error eliminando: ' + (e.message || JSON.stringify(e)
));

                                    }
                                }}>Eliminar</button>
                            </div>
                        </div>
                    </li>
                ))}
            </ul>
        </div>
    );
};

export default ArticleList;
```

== File: src/components/ArticleList/ArticleList.css ==

```
.article-list {
    /* Estilo simple del contenedor */
    padding: 1rem;
}
.article-item {
    border: 1px solid #ddd;
    padding: 0.75rem;
    margin-bottom: 0.5rem;
    border-radius: 6px;
}
.article-item h3 {
    margin: 0 0 0.25rem 0;
}
.article-item .actions {
    margin-top: 0.5rem;
}
.article-item .actions button {
    margin-right: 0.5rem;
}


== File: src/components/ArticleForm/ArticleForm.jsx ==

import React, { useState, useEffect } from 'react'; // Import React y hooks necesarios
import './ArticleForm.css'; // Importar estilos

// Formulario para crear/editar artículos. Recibe dispatch, estado y acciones por props.
const ArticleForm = ({ state, dispatch, actions, services }) => {
    // useState para controlar campos del formulario
    const [title, setTitle] = useState(''); // Título del artículo
    const [content, setContent] = useState(''); // Contenido del artículo
    const [saving, setSaving] = useState(false); // Indicador de guardado

    // useEffect para rellenar el formulario cuando editing cambia
    useEffect(() => {
        if(state.editing) {
            setTitle(state.editing.title || '');
            setContent(state.editing.content || '');
        } else {
            setTitle('');
            setContent('');
        }
    }, [state.editing]);

    // Función para guardar el formulario (async/await)
    const handleSubmit = async (e) => {
        e.preventDefault(); // Evitar comportamiento por defecto del form
        setSaving(true); // Poner indicador de guardado
        try {
            // Si hay editing llamamos a update, si no a create
            if(state.editing) {
                const updated = await services.updateArticleById(state.editing._id, { title, content
 });
                // Dispatch acción de actualización con el artículo devuelto
                dispatch({ type: actions.UPDATE_ARTICLE, payload: updated });
            } else {
                const created = await services.createArticle({ title, content });
                // Dispatch para añadir el nuevo artículo
                dispatch({ type: actions.ADD_ARTICLE, payload: created });
```

```
            }
            // Limpiar formulario y edición
            dispatch({ type: actions.CLEAR_EDITING });
            setTitle('');
            setContent('');
        } catch (err) {
            // Mostrar error al usuario
            alert('Error guardando artículo: ' + (err.message || JSON.stringify(err)));
        } finally {
            setSaving(false); // Apagar indicador de guardado
        }
    };

    // Render del formulario
    return (
        <form className='article-form' onSubmit={handleSubmit}>
            <h2>{state.editing ? 'Editar artículo' : 'Crear artículo'}</h2>
            <label>
                Título
                <input value={title} onChange={(e) => setTitle(e.target.value)} required />
            </label>
            <label>
                Contenido
                <textarea value={content} onChange={(e) => setContent(e.target.value)} required />
            </label>
            <div className='buttons'>
                <button type='submit' disabled={saving}>{saving ? 'Guardando...' : 'Guardar'}</butto
n>
                {state.editing && (
                    <button type='button' onClick={() => dispatch({ type: actions.CLEAR_EDITING })}>
Cancelar</button>
                )}
            </div>
        </form>
    );
};

export default ArticleForm;


== File: src/components/ArticleForm/ArticleForm.css ==

.article-form {
    padding: 1rem;
    border: 1px solid #eee;
    border-radius: 6px;
}
.article-form label {
    display: block;
    margin-bottom: 0.5rem;
}
.article-form input, .article-form textarea {
    width: 100%;
    padding: 0.5rem;
    margin-top: 0.25rem;
    box-sizing: border-box;
}
.article-form .buttons {
    margin-top: 0.5rem;
}
```

```
.article-form .buttons button {
    margin-right: 0.5rem;
}


== File: src/App.jsx ==

import React, { useReducer } from 'react'; // Import React y useReducer
import './App.css'; // Import styles globales
import ArticleList from './components/ArticleList/ArticleList.jsx'; // Componente lista
import ArticleForm from './components/ArticleForm/ArticleForm.jsx'; // Componente formulario
import { articleReducer } from './state/article.reducer.js'; // Reducer
import { ACTIONS } from './state/article.actions.js'; // Acciones
import * as services from './services/articleService.js'; // Servicios de API

// Estado inicial para el reducer
const initialState = {
    articles: [], // Array de artículos
    loading: false, // Indicador de carga
    error: null, // Error en carga
    editing: null // Artículo en edición
};

// Componente principal de la aplicación
const App = () => {
    // Uso avanzado del hook useReducer para manejar el estado complejo
    const [state, dispatch] = useReducer(articleReducer, initialState);

    // Render principal: pasamos state, dispatch, acciones y servicios a los componentes hijos
    return (
        <div className='app'>
            <header className='app-header'>
                <h1>Gestión de Artículos</h1>
            </header>
            <main className='app-main'>
                {/* Columna de formulario */}
                <section className='left'>
                    <ArticleForm state={state} dispatch={dispatch} actions={ACTIONS} services={servi
ces} />
                </section>
                {/* Columna de lista */}
                <section className='right'>
                    <ArticleList state={state} dispatch={dispatch} actions={ACTIONS} services={servi
ces} />
                </section>
            </main>
        </div>
    );
};

export default App;


== File: src/App.css ==

.app {
    font-family: Arial, sans-serif;
    max-width: 1000px;
    margin: 0 auto;
    padding: 1rem;
```

```
}
.app-header {
    text-align: center;
    margin-bottom: 1rem;
}
.app-main {
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap: 1rem;
}


== File: src/main.jsx ==

import React from 'react'; // Import React
import { createRoot } from 'react-dom/client'; // Import función para montar la app
import App from './App.jsx'; // Importar componente App

// Montaje de la aplicación en el elemento con id 'root'
const container = document.getElementById('root');
const root = createRoot(container);
root.render(<App />);


== Explicaciones teóricas (hooks, componentes, patrón useReducer) ==


useReducer:
- useReducer es un hook de React que permite manejar estado complejo o con múltiples sub-valores de
forma predecible.
- Se usa proporcionando un reducer (función que recibe state y action) y un estado inicial.
- Es útil cuando se tienen muchas acciones que transforman el estado o cuando la lógica de actualiza
ción es complicada.

useEffect:
- useEffect permite ejecutar efectos secundarios (cargas de datos, suscripciones) al montar o cuando
 cambian dependencias.
- En ArticleList usamos useEffect para cargar los artículos al montar el componente.

useState:
- useState gestiona estado local simple (por ejemplo, campos del formulario).
- En ArticleForm usamos useState para controlar título, contenido y estado de guardado.

Patrón general:
- Centralizamos la lógica de estado en un reducer (article.reducer.js) y definimos tipos de acciones
 en article.actions.js.
- Las llamadas a la API se encapsulan en services/articleService.js que usa la función genérica API.
js proporcionada.
- Los componentes son puros en React (funcionales con arrow functions) y reciben estado y dispatch c
omo props para interactuar con el store local.
- Los archivos CSS están separados por componente y colocados en el mismo directorio.

Comentarios en línea:
- Cada archivo contiene comentarios en cada línea o bloque importante explicando su propósito y su f
uncionamiento.
```