// Определяем тип для продукта

interface Product {

id: number; // Уникальный идентификатор продукта

name: string; // Название продукта

}

// Функция для получения данных с сервера

const fetchProducts = async (): Promise<Product[]> => {

try {

// URL эндпоинта API

const response = await fetch('https://api.example.com/products');

if (!response.ok) {

throw new Error('Ошибка при запросе данных с сервера');

}

const data: Product[] = await response.json(); // Типизированные данные

return data;

} catch (error) {

console.error('Ошибка:', error);

return []; // Возвращаем пустой массив при ошибке

}

};

// Функциональный компонент со списком продуктов

const ProductsList: React.FC = () => {

const [products, setProducts] = useState<Product[]>([]); // Стейт для продуктов

const [loading, setLoading] = useState<boolean>(true); // Стейт для статуса загрузки

const [error, setError] = useState<string | null>(null); // Стейт для обработки ошибок

// Хук useEffect вызывается при монтировании компонента

useEffect(() => {

const fetchData = async () => {

setLoading(true); // Устанавливаем статус загрузки

try {

const data = await fetchProducts(); // Вызываем функцию для загрузки данных

setProducts(data); // Устанавливаем полученные данные в стейт

} catch (err) {

setError((err as Error).message); // Сохраняем сообщение ошибки (преобразуем объект ошибки в тип Error)

} finally {

setLoading(false); // Отключаем статус загрузки

}

};

fetchData();

}, []); // [] - зависимости, хук вызывается только один раз (при монтировании компонента)

// Пока данные загружаются

if (loading) {

return <p>Загрузка...</p>;

}

// Если произошла ошибка

if (error) {

return <p>Ошибка: {error}</p>;

}

// Если данные успешно загружены

return (

<div>

<h1>Список продуктов</h1>

<ul>

{products.map((product) => (

<li key={product.id}>{product.name}</li> // Отображение продуктов в списке

))}

</ul>

</div>

);

};