import axios from 'axios';

const API\_URL = 'http://localhost:3000';

// Получение данных о текущем пользователе

export async function fetchMe(): Promise<UserMeResponse> {

try {

const response = await axios.get<UserMeResponse>(`${API\_URL}/me`, {

withCredentials: true,

headers: { 'Content-Type': 'application/json' }

});

return response.data;

} catch (error) {

throw new Error('Ошибка при получении данных');

}

}

// Регистрация нового пользователя

export async function register({ name, surname, email, password }: RegisterType): Promise<void> {

try {

await axios.post(`${API\_URL}/register`,

{ name, surname, email, password },

{ withCredentials: true, headers: { 'Content-Type': 'application/json' } }

);

} catch (error) {

throw error;

}

}

// Логин пользователя

export async function login({ email, password }: LoginType): Promise<void> {

try {

await axios.post(`${API\_URL}/login`,

{ email, password },

{ withCredentials: true, headers: { 'Content-Type': 'application/json' } }

);

} catch (error) {

throw error;

}

}

import axios from 'axios';

const API\_URL = 'http://localhost:3000';

const apiClient = axios.create({

baseURL: API\_URL,

withCredentials: true,

headers: {

'Content-Type': 'application/json',

},

});

// Получение данных о текущем пользователе

export async function fetchMe(): Promise<UserMeResponse> {

try {

const response = await apiClient.get<UserMeResponse>('/me');

return response.data;

} catch (error) {

throw new Error('Ошибка при получении данных');

}

}

// Регистрация нового пользователя

export async function register({ name, surname, email, password }: RegisterType): Promise<void> {

try {

await apiClient.post('/register', { name, surname, email, password });

} catch (error) {

// Здесь можно добавить кастомную обработку ошибок, если требуется

throw error;

}

}

// Логин пользователя

export async function login({ email, password }: LoginType): Promise<void> {

try {

await apiClient.post('/login', { email, password });

} catch (error) {

throw error;

}

}

Вот ключевые преимущества \*\*Axios\*\* по сравнению со стандартным `fetch` в JavaScript:

---

### 1. Простая обработка JSON

- \*\*Axios\*\* автоматически преобразует ответ в JSON, если это возможно (`response.data` сразу объект).

- \*\*fetch:\*\* нужно вручную вызывать `response.json()` и следить за статусом ответа.

---

### 2. Глобальная и индивидуальная настройка (экземпляры)

- В \*\*Axios\*\* можно создавать отдельные экземпляры с базовыми настройками (`baseURL`, заголовки, авторизация).

- \*\*fetch\*\* не поддерживает глобальные настройки — всё передается вручную в каждом запросе.

---

### 3. Обработка HTTP-ошибок

- В \*\*Axios\*\* любые коды ответов, кроме 2xx, считаются ошибкой и попадают в `.catch`.

- \*\*fetch\*\* бросает ошибку только при сетевых сбоях, а для статусов типа 401/404/500 — не бросает; нужно самим проверять `response.ok`.

---

### 4. Поддержка `withCredentials`

- В \*\*Axios\*\* достаточно указать `withCredentials: true` для работы с cookie.

- В \*\*fetch\*\* аналог — `credentials: 'include'`, но реализация чуть сложнее в некоторых случаях.

---

### 5. Сериализация данных для форм и URL

- \*\*Axios\*\* сам преобразует объекты в строку запроса для GET или в `FormData` при отправке файлов.

- \*\*fetch\*\* требует самостоятельно сериализовать параметры.

---

### 6. Интерцепторы

- В \*\*Axios\*\* можно легко добавлять перехватчики (interceptors) для модификации запроса или обработки ответа/ошибок централизовано.

- \*\*fetch\*\* не поддерживает эту функцию ― приходится оборачивать вызовы вручную.

---

### 7. Отмена запросов

- В \*\*Axios\*\* есть удобная поддержка отмены запросов через `AbortController`.

- \*\*fetch\*\* поддерживает отмену только с нового стандарта, и реализовать отмену сложнее для начинающих.

---

### 8. Поддержка старых браузеров

- \*\*Axios\*\* работает даже в Internet Explorer.

- \*\*fetch\*\* официально не поддерживается в IE без полифиллов.

---

### 9. Обработка таймаутов

- В \*\*Axios\*\* можно задать таймаут прямо в настройках клиента.

- В \*\*fetch\*\* таймаут нужно реализовывать вручную через сторонние решения (например, через `Promise.race`).

---

## Кратко

\*\*Axios\*\*:

- Более дружелюбен к разработчику;

- Меньше “ручной” обработки ошибок и данных;

- Есть расширенная функциональность “из коробки”.

\*\*fetch\*\*:

- Легче и без сторонних зависимостей;

- Хорош для простых сценариев.