

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Фронтэнд-разработка

**Отчет**

**Лабораторная работа №2**

**Выполнил:**

**Гнеушев Владислав**

**К3439**

**Проверил:  
Добряков Д. И.**

**Санкт-Петербург**

**2025 г.**

## Задача

Вам нужно привязать то, что Вы делали в ЛР1 к внешнему API средствами fetch/axios/xhr. Реализуйте моковое API средствами JSON-сервера и подключите к нему авторизацию, как в примерах, которые мы рассматривали в рамках тем "Имитация работы с API".

**Выбранный вариант - Платформа для фитнес-тренировок и здоровья**

## Ход работы

### 1. Настройка json-сервера

Были заполнены данные файла db.json при помощи LLM.



The screenshot shows a code editor with several tabs at the top: api.js, package.json, db.json (which is the active tab), blog.js, and workouts.js. The db.json tab displays the following JSON data:

```
1  {
2    "users": [
3      {
4        "id": 1,
5        "name": "Анна Петрова",
6        "email": "anna@example.com",
7        "password": "secret123",
8        "stats": {
9          "completedWorkouts": 18,
10         "burnedCalories": 4200,
11         "workoutDays": 24,
12         "currentStreak": 5
13       },
14       "plan": {
15         "monday": {
16           "workoutId": 2,
17           "title": "Силовая тренировка для рук",
18           "duration": 45,
19           "status": "scheduled"
20         },
21         "tuesday": {
22           "workoutId": 3,
23           "title": "Йога для начинающих",
24           "duration": 30,
25           "status": "scheduled"
26         },
27         "wednesday": {
28           "workoutId": 2,
29           "title": "Силовая тренировка для рук",
30           "duration": 45,
31           "status": "completed"
32         },
33         "thursday": null,
34         "friday": {
35           "workoutId": 3,
36           "title": "Йога для начинающих",
37           "duration": 30,
38           "status": "scheduled"
39         },
40         "saturday": null,
41         "sunday": null
42     }
43   }
```

Рисунок 1 — Содержание файла db.json

### 2. Рефакторинг JS-кода

С 1 лабораторной работы оставался большой файл [main.js](#), который был разбит на несколько файлов с разной ответственностью.

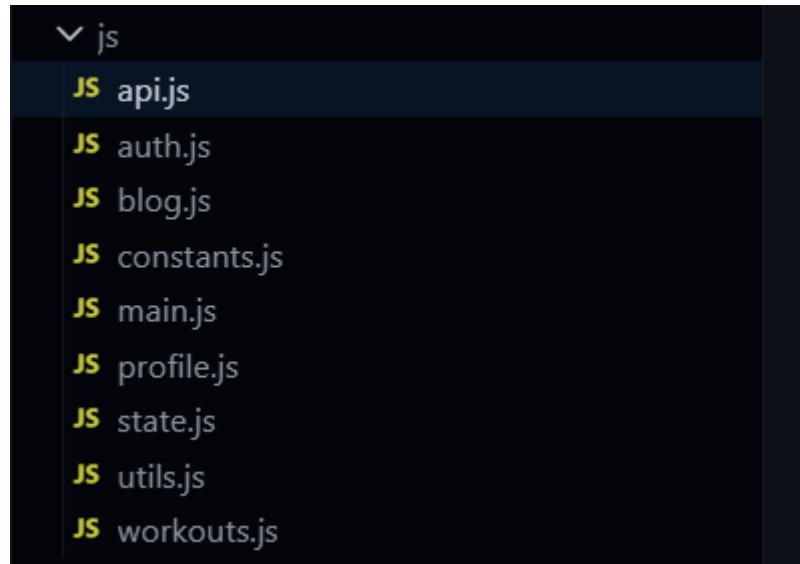


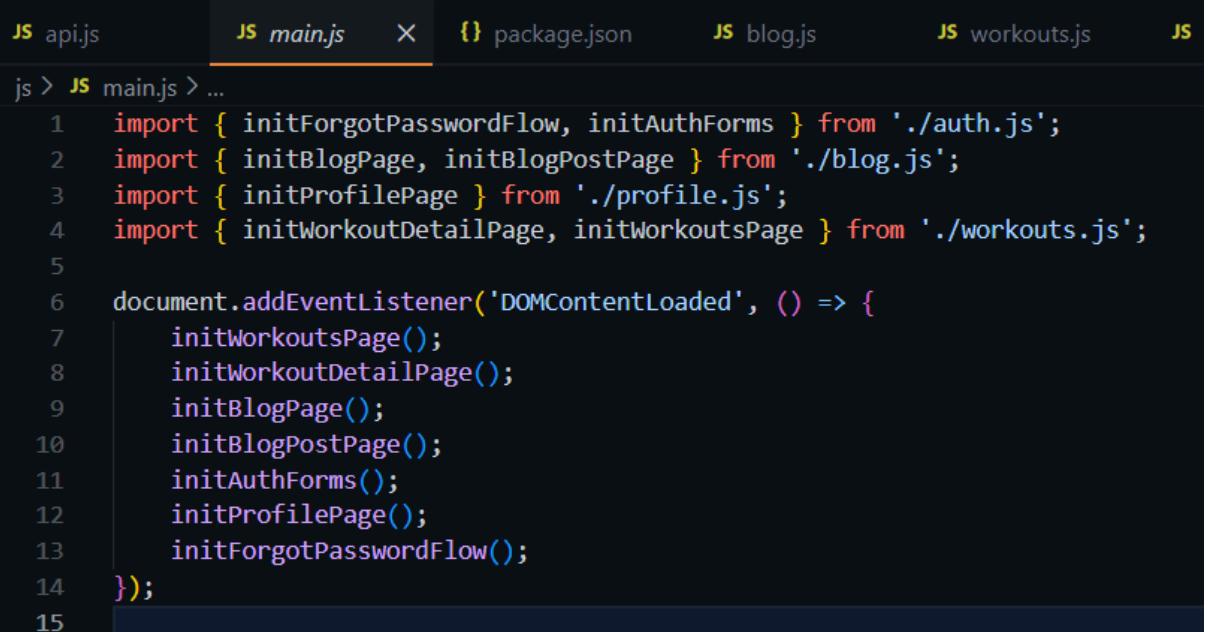
Рисунок 2 — Рефакторинг большого файла main.js

Помимо разбиения существующего функционала, был добавлен модуль [api.js](#) - он содержит функции, делающие запросы к API (json-server).

```
js > JS api.js > fetchBlogPosts
1 import { API_BASE_URL } from './constants.js';
2 import { state, persistUser } from './state.js';
3 import { buildEmptyPlan, buildEmptyStats } from './utils.js';
4
5 export async function fetchWorkouts() {
6     const response = await fetch(`${API_BASE_URL}/workouts`);
7     if (!response.ok) {
8         throw new Error('Не удалось получить список тренировок');
9     }
10    return response.json();
11 }
12
13 export async function fetchWorkoutById(id) {
14     const response = await fetch(`${API_BASE_URL}/workouts/${id}`);
15     if (!response.ok) {
16         throw new Error('Не удалось получить тренировку');
17     }
18    return response.json();
19 }
20
21 export async function fetchBlogPosts() {
22     const response = await fetch(`${API_BASE_URL}/blogPosts`);
23     if (!response.ok) {
24         throw new Error('Не удалось получить статьи');
25     }
26    return response.json();
```

Рисунок 3 — Запросы к API при помощи fetch

А `main.js` инициализирует страницы сразу после загрузки DOM-дерева страницы.



```
js api.js          JS main.js X  {} package.json    JS blog.js      JS workouts.js JS
js > JS main.js > ...
1 import { initForgotPasswordFlow, initAuthForms } from './auth.js';
2 import { initBlogPage, initBlogPostPage } from './blog.js';
3 import { initProfilePage } from './profile.js';
4 import { initWorkoutDetailPage, initWorkoutsPage } from './workouts.js';
5
6 document.addEventListener('DOMContentLoaded', () => {
7     initWorkoutsPage();
8     initWorkoutDetailPage();
9     initBlogPage();
10    initBlogPostPage();
11    initAuthForms();
12    initProfilePage();
13    initForgotPasswordFlow();
14 });
15
```

Рисунок 4 — Инициализация страниц

## Вывод

В результате выполнения лабораторной работы в проект были добавлены вызовы API-эндпоинтов json-server для эмуляции запросов к реальному API-серверу. Также был разбит большой javascript-файл на несколько маленьких, имеющих разные зоны ответственности.