

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Фронт-энд разработка

**Отчет**

**Лабораторная работа №2**

**Выполнила:**

**Стукалова Александра**

**Группа К3343**

**Проверил:  
Добряков Д. И.**

**Санкт-Петербург**

**2026 г**

## **Введение**

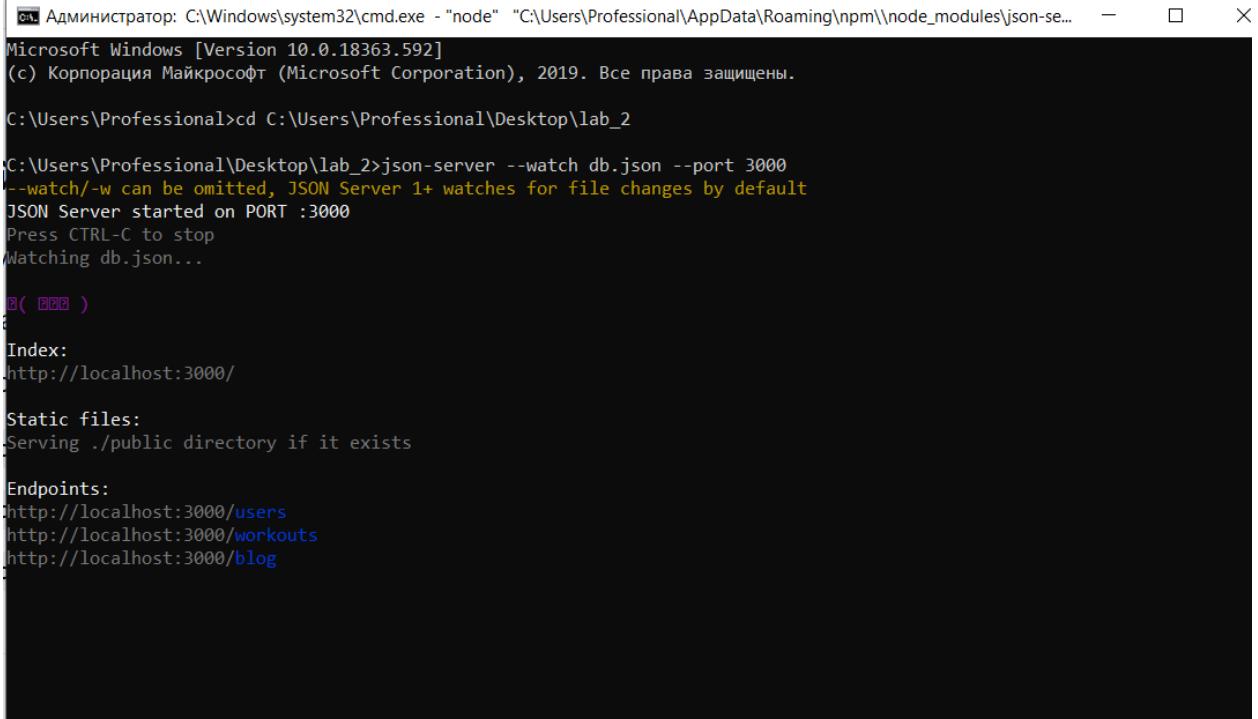
В данной лабораторной работе нужно было подключить сайт из первой лабораторной к внешнему API. В качестве API я использовала JSON-server - инструмент, который позволяет быстро создать фейковый сервер на основе JSON-файла.

### **Задачи данной лабораторной работы:**

- Установить и запустить JSON-server;
- Создать файл db.json с данными (пользователи, тренировки, статьи);
- Переделать страницы так, чтобы они загружали данные с сервера через fetch;
- Реализовать авторизацию, чтобы проверять пользователя при входе;
- Реализовать регистрацию, чтобы добавлять новых пользователей в базу.

## Содержимое работы:

Для работы с данными я использовала JSON-server, который позволяет быстро создать моковое API на основе JSON-файла. Сервер запускается локально на компьютере и работает на порту 3000:



```
Administrator: C:\Windows\system32\cmd.exe - "node" "C:\Users\Professional\AppData\Roaming\npm\node_modules\json-se... — X
Microsoft Windows [Version 10.0.18363.592]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\Professional>cd C:\Users\Professional\Desktop\lab_2

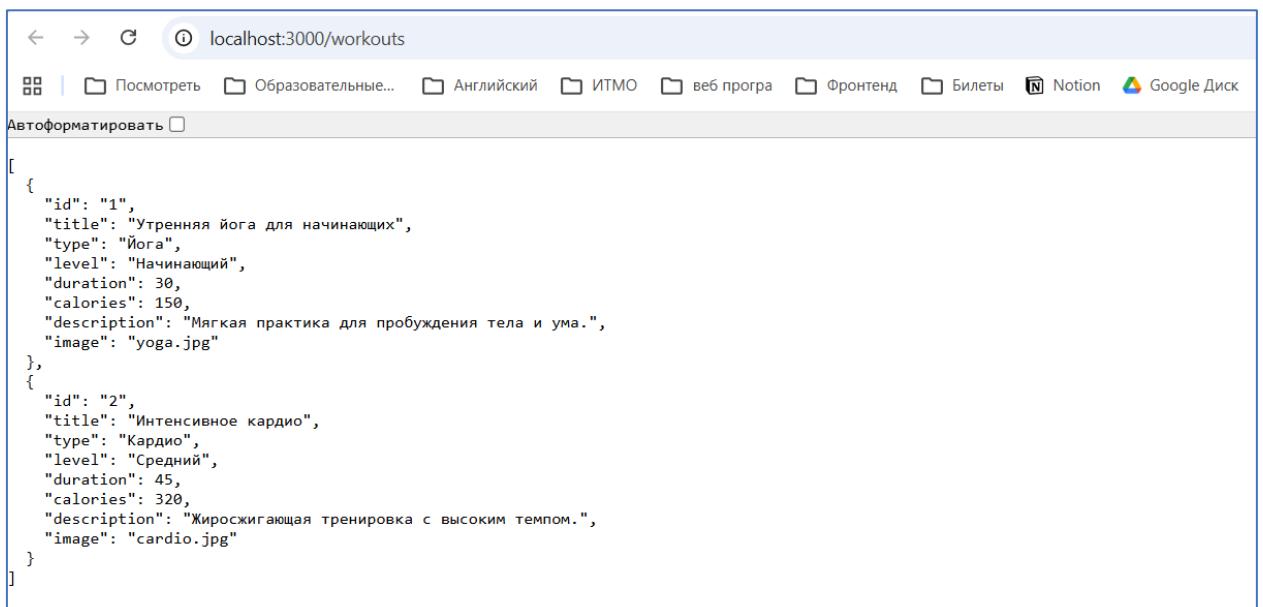
C:\Users\Professional\Desktop\lab_2>json-server --watch db.json --port 3000
--watch/-w can be omitted, JSON Server 1+ watches for file changes by default
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching db.json...

B( 000 )
:
Index:
http://localhost:3000/

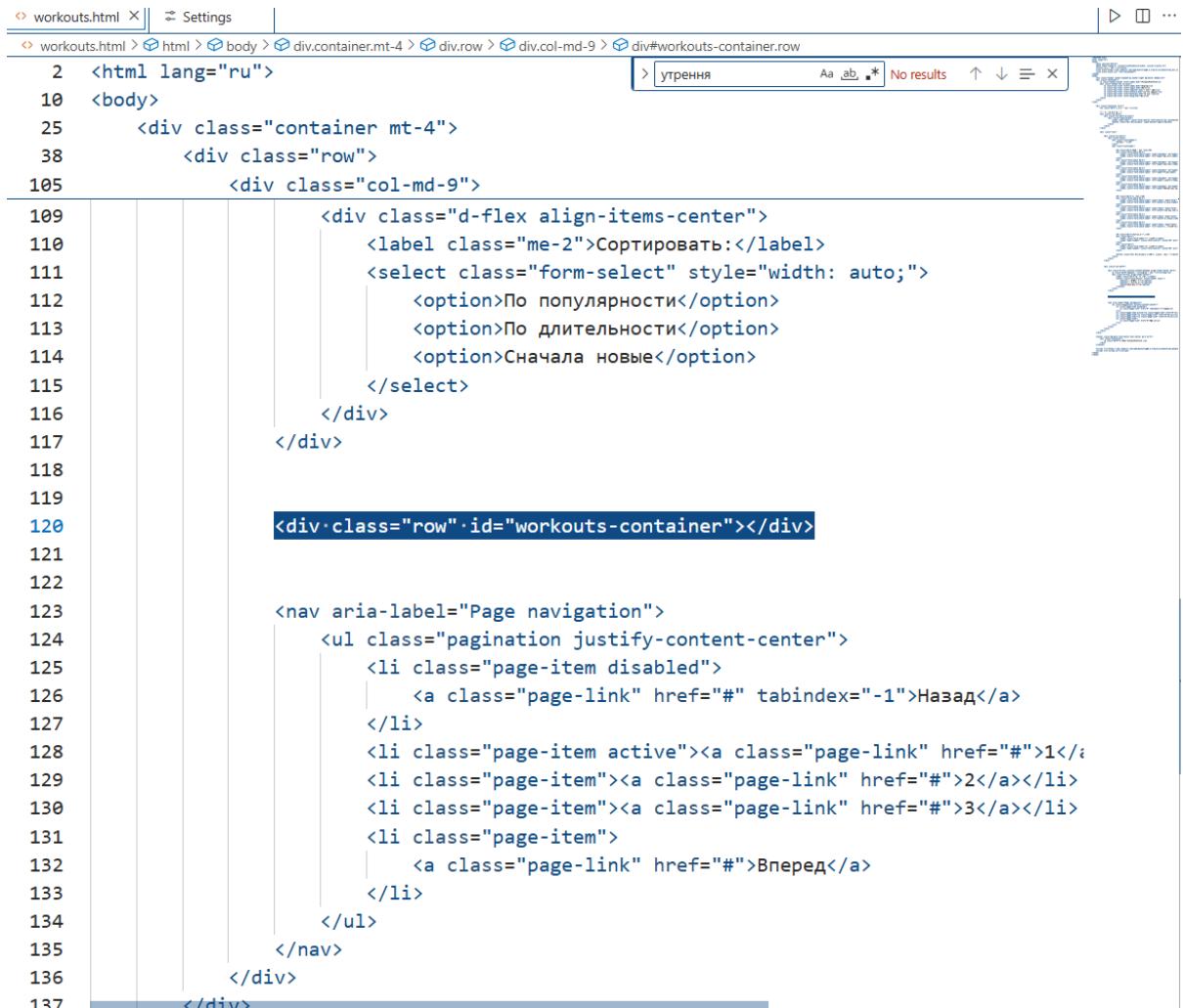
Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/users
http://localhost:3000/workouts
http://localhost:3000/blog
```

Сервер стал доступен по адресу <http://localhost:3000>. Все страницы сайта теперь обращаются к этому серверу через fetch и получают данные в реальном времени:



Для того, чтобы тренировки загружались с сервера (из файла db.json через JSON-server) я в workouts удалила статичные карточки (раньше в коде было 4 готовые карточки с йогой, кардио, силовой и растяжкой) и добавила пустой контейнер. Это место, куда JavaScript будет вставлять карточки:



```
2 <html lang="ru">
10 <body>
25   <div class="container mt-4">
38     <div class="row">
105       <div class="col-md-9">
109         <div class="d-flex align-items-center">
110           <label class="me-2">Сортировать:</label>
111           <select class="form-select" style="width: auto;">
112             <option>По популярности</option>
113             <option>По длительности</option>
114             <option>Сначала новые</option>
115           </select>
116         </div>
117       </div>
118
119
120       <div class="row" id="workouts-container"></div>
121
122
123       <nav aria-label="Page navigation">
124         <ul class="pagination justify-content-center">
125           <li class="page-item disabled">
126             <a class="page-link" href="#" tabindex="-1">Назад</a>
127           </li>
128           <li class="page-item active"><a class="page-link" href="#">1</a>
129           <li class="page-item"><a class="page-link" href="#">2</a></li>
130           <li class="page-item"><a class="page-link" href="#">3</a></li>
131           <li class="page-item">
132             <a class="page-link" href="#">Вперед</a>
133           </li>
134         </ul>
135       </nav>
136     </div>
137   </div>
```

Также я написала код в script.js, чтобы он забирал данные с сервера по адресу. При загрузке страницы JS отправляет запрос на сервер

```
fetch('http://localhost:3000/workouts')
```

Сервер возвращает список тренировок из файла db.json. Для каждой тренировки JS создает карточку с картинкой, заголовком, описанием и кнопкой. Карточки вставляются в пустой контейнер. Количество найденных тренировок теперь тоже обновляется автоматически. Вместо числа 12 подставляется реальное количество записей с сервера:

## ПОИСК ТРЕНИРОВОК

Например: йога для начинающих

Найти

Найдено: 2

Сортировать: По популярности ▾

Фильтры

Тип тренировки

Кардио

Силовая

Йога

Пилатес

Функциональная

Уровень

Любой

Начинающий

Средний

Продвинутый

Длительность

От (минут)

10



Утренняя йога для начинающих  
Мягкая практика для пробуждения тела и ума.

Йога Начинающий 30 мин

Смотреть



Интенсивное кардио  
Жироожигающая тренировка с высоким темпом.

Кардио Средний 45 мин

Смотреть

Назад 1 2 3 Вперед

Далее на странице блога я заменила статичные карточки статей на динамические. Раньше в HTML было прописано 6 статей, теперь я убрала их

и добавила пустой контейнер <div id="blog-container">

```
2  <html lang="ru">
10 <body>
24   <div class="container mt-4">
38     <div class="row mb-4">
39       <div class="col-12">
41         <button class="btn btn-outline-primary active">Все</button>
42         <button class="btn btn-outline-primary">Питание</button>
43         <button class="btn btn-outline-primary">Тренировки</button>
44         <button class="btn btn-outline-primary">Здоровье</button>
45         <button class="btn btn-outline-primary">Рецепты</button>
46       </div>
47     </div>
48   </div>
49
50   <!-- Статьи -->
51   <div class="row" id="blog-container"></div>
52
53   <!-- Пагинация -->
54   <nav aria-label="Page navigation" class="mt-4">
55     <ul class="pagination justify-content-center">
56       <li class="page-item disabled">
57         <a class="page-link" href="#">Назад</a>
58       </li>
59       <li class="page-item active"><a class="page-link" href="#">1</a></li>
60       <li class="page-item"><a class="page-link" href="#">2</a></li>
61       <li class="page-item"><a class="page-link" href="#">3</a></li>
62       <li class="page-item">
63         <a class="page-link" href="#">Вперед</a>
64       </li>
65     </ul>
66   </nav>
67 </div>
68
69 <footer class="bg-white text-center py-3 mt-5 shadow-sm">
```

В файл db.json добавила данные о трех статьях (про ПП-завтраки, воду и кардио), и теперь JS загружает их с сервера при открытии страницы:

The screenshot shows a web browser window with the URL [C:/Users/Professional/Desktop/lab\\_2/blog.html](C:/Users/Professional/Desktop/lab_2/blog.html). The page is titled "FitnessPlatform" and features a header with "Блог о здоровье и питании". Below the header is a search bar with the placeholder "Поиск статей..." and a "Найти" button. A navigation menu includes "Все", "Питание", "Тренировки", "Здоровье", and "Рецепты". The main content area displays three blog posts in a grid:

- 5 простых ПП-завтраков**  
Быстрые и полезные рецепты для идеального начала дня.  
Питание | 15 мин | Читать
- Сколько воды нужно пить?**  
Разбираемся в нормах и мифах о водном балансе.  
Здоровье | 5 мин | Читать
- Кардио для начинающих**  
Как начать бегать и не бросить через неделю.  
Тренировки | 7 мин | Читать

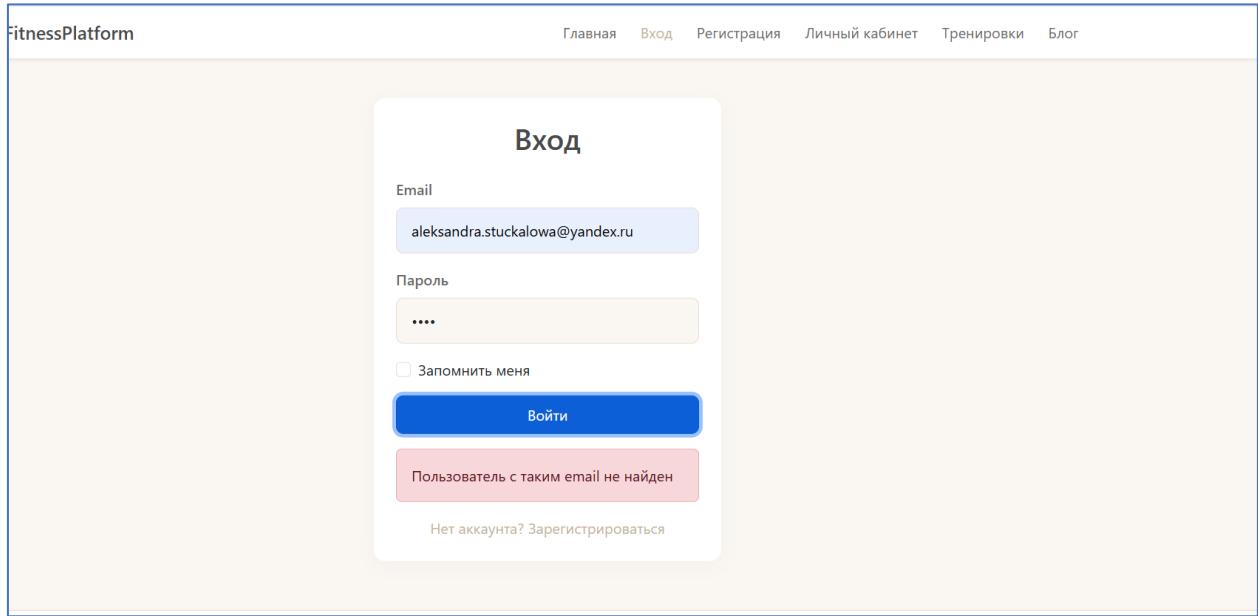
At the bottom of the page, there are navigation links for "Назад", "1", "2", "3", and "Вперед". The footer contains the copyright notice "© 2026 FitnessPlatform".

Далее на странице входа я добавила возможность проверять пользователя по данным из JSON-сервера. Для проверки в db.json добавлен пользователь с email s.stukalova@example.com и паролем 123456.

Теперь при нажатии на кнопку «Войти» отправляется запрос на сервер, идет поиск пользователя с введенным email и сравнивается пароль. Если данные совпадают, появляется сообщение об успешном входе и открывается модальное окно с приветствием:

The screenshot shows the "FitnessPlatform" login page. A modal window is displayed with the message "Добро пожаловать!" and "Вы успешно вошли в FitnessPlatform!". It also asks "Готовы к тренировке?". A large "Начать" button is visible at the bottom of the modal. The background shows the login form fields: "Пароль" (Password) with an input field containing "Ведите пароль" (Enter password), a "Запомнить меня" (Remember me) checkbox, and a "Войти" (Login) button. Below the button, a green success message says "Вход выполнен успешно!". At the bottom of the page, there is a link "Нет аккаунта? Зарегистрироваться" (No account? Register) and the footer "© 2025 FitnessPlatform".

Если пользователь не найден или пароль неверный, показывается красное сообщение с ошибкой:

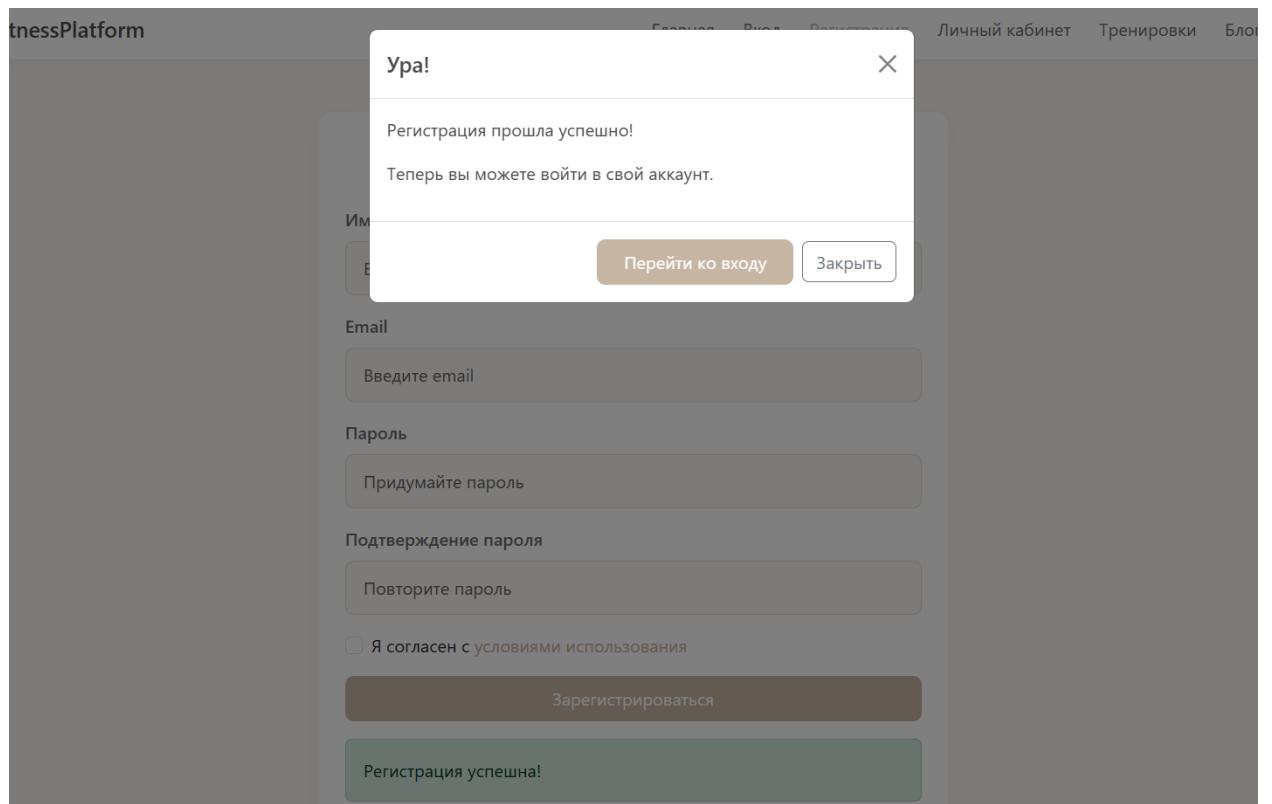


Далее я работала со страницей регистрации. При заполнении формы проверяется, совпадают ли пароли, и отправляет запрос на сервер, чтобы убедиться, что введенный email еще не занят. Если все хорошо, отправляется запрос с данными нового пользователя, и тот сохраняется в db.json:

A screenshot of a JSON editor showing the file "db.json". The content is a JSON object with a "users" array containing two user entries. The first user has an id of 1, name "Александра Стукалова", email "s.stukalova@example.com", and password "123456". The second user has an id of 1771368374853, name "Александра Стукалова", email "aleksandra.stuckalowa@yandex.ru", and password "123".

```
1 {  
2   "users": [  
3     {  
4       "id": 1,  
5       "name": "Александра Стукалова",  
6       "email": "s.stukalova@example.com",  
7       "password": "123456"  
8     },  
9     {  
10       "id": 1771368374853,  
11       "name": "Александра Стукалова",  
12       "email": "aleksandra.stuckalowa@yandex.ru",  
13       "password": "123"  
14     }  
15   ],  
16 }
```

После успешной регистрации появляется сообщение и модальное окно с предложением перейти ко входу:



Если email уже существует или пароли не совпадают, показываются красные сообщения с ошибкой:

