

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Ананьев Никита

Группа К3440

**Проверил:
Добряков Д. И.**

Санкт-Петербург

2025 г.

Задача

Привязать то, что было сделано в рамках ЛР1 к внешнему API средствами fetch/axios/xhr. Реализовать моковое API средствами JSON-сервера и подключить к нему авторизацию, как в примерах, рассматриваемых в рамках тем "Имитация работы с API".

Ход работы

В качестве мокового API был использован npm пакет json-server. На рисунке 1 показана команда выполнения запуска json сервера. Для данной лабораторной в db.json (хранилище, используемое сервером) добавлены две коллекции - users и properties. Пример данных, которые сохраняются в эти коллекции см. на рисунке 2.

```
PS C:\Users\Nikita\ITMO-ACS-Frontend-2025\labs\К3340\Ананьев Никита\lr2> npm start

> start
> npx json-server --watch db.json

--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...

( u_wu )

Index:
http://localhost:3000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/users
http://localhost:3000/properties
```

Рисунок 1 – Запуск json сервера

```
{  
    "id": "cc22",  
    "name": "Тест",  
    "email": "just@man.ru",  
    "phone": "+7 (988) 888-88-56",  
    "password": "777",  
    "role": "tenant"  
}  
],  
"properties": [  
    {  
        "id": "1",  
        "owner_id": 1,  
        "type": "flat",  
        "title": "Квартира",  
        "description": "Aaa",  
        "price": 80000,  
        "is_daily_payment": false,  
        "address": "Санкт-Петербург, Невский пр., 88",  
        "feed_img": "../media/images/nevskiy.jpg",  
        "images": [  
            "../media/images/nevskiy.jpg"  
        ],  
        "rooms": 2,  
        "floor": 2,  
        "area": 60  
    },  
]
```

Рисунок 2 – Пример данных в db.json

Авторизация

Теперь при регистрации сначала происходит проверка на существование пользователя в системе (идентификация) по email. Если пользователь проходит идентификацию, регистрация невозможна и предлагается войти в систему под имеющимся аккаунтом (см. рисунок 3 и 4).

```
async function identifyUserByEmail(email) {  
    let exists = false;  
  
    await fetch(`${USERS_URL}?email=${email}`)  
        .then((response) => {  
            return response.json();  
        })  
        .then((data) => {  
            exists = (data.length > 0) ? true : false;  
        })  
  
    return exists;  
}
```

Рисунок 3 – Функция идентификации пользователя

```
const exists = await identifyUserByEmail(email);
if (exists) {
    alert('Пользователь с таким email уже зарегистрирован');
    return;
}

try {
    const response = await fetch(
        USERS_URL, {
            method: "POST",
            headers: {
                "Content-Type": "application/json; charset=utf-8",
            },
            body: JSON.stringify({name, email, phone, password, role: DEFAULT_ROLE})
        }
    );

    if (!response.ok)
        throw new Error(`HTTP error, status: ${response.status}`);

    window.location.href = 'login.html';
} catch (e) {
    console.error(e);
    alert('Ошибка регистрации');
}
```

Рисунок 4 – Регистрация (создание пользователя)

В процессе авторизации происходит запрос к json серверу на получение данных пользователя, далее идет сравнение пароля из базы с введенным в форму. В случае успешного входа данные о пользователе сохраняются в localStorage и затем уже забираются оттуда по мере необходимости.

Главная страница

Список объектов недвижимости загружается запросом к json-серверу с набором query-параметров, которые формируются на основе применяемых пользователем фильтров. Фильтры сохраняются в карту filters и при формировании запроса применяется функция buildQuery (см. рисунок 5).

```

1  function buildQuery(filters) {
2      if (filters.size === 0) return "";
3
4      const params = new URLSearchParams();
5
6      filters.forEach((value, key) => {
7          if (key === DEFAULT_SELECT)
8              return;
9
10         if (key === "price") [
11             const [min, max] = value.split("-");
12             params.append("price_gte", min);
13             params.append("price_lte", max);
14         ]
15         else if (Array.isArray(value)) {
16             value.forEach(v => params.append(key, v));
17         } else {
18             params.append(key, value);
19         }
20     });
21
22     return "?" + params.toString();
23 }

```

Рисунок 5 – Функция построения query параметров запроса

Когда сервер возвращает отфильтрованные записи, блоки карточек программно добавляются на html страницу (см. рисунок 6, 7 и 8).

```

async function filterProperties() {
    collectFilters();
    const query = buildQuery(filters);

    response = await fetch(PROPERTIES_URL + query);

    if (!response.ok) {
        console.error(`Failed to load properties list; response status = ${response.status}`);
        return;
    }

    data = await response.json();
    const cards = document.getElementById("cards");
    cards.innerHTML = "";

    if (data.length == 0) {
        return;
    }

    data.forEach(prop => {
        cards.append(buildCardHTML(prop));
    });
}

window.scrollTo(0, 0);

```

Рисунок 6 – Получение объектов недвижимости

```

function buildCardHTML(property) {
    const newCard = document.createElement("div");
    newCard.className = "card";

    const img = document.createElement("img");
    img.src = (property.feed_img !== undefined) ? property.feed_img : DEFAULT_PICTURE_URL;

    const info = document.createElement("div");
    info.className = "card-info";

    const price = document.createElement("div");
    price.className = "price";
    price.innerHTML = `${property.price} ₽ / `;
    let frequency = "месяц";

    if (property.is_daily_payment)
        frequency = "день";

    price.innerHTML += frequency;

    const link = document.createElement("div");
    link.innerHTML = `2, \${property.rooms} комнат\(-ы\)`;
    if \(property.floor !== undefined\)
        description.innerHTML += `, \${property.floor}-й этаж`;
    description.innerHTML += `  
\${property.description}`;

    info.append\(price, link, description\);
    newCard.append\(img, info\);

    return newCard;
}

```

Рисунок 7 – Формирование html для карточки

The screenshot shows a search interface for real estate. On the left, there is a sidebar with various filters:

- Метро, адрес или ЖД станция...**: A search input field.
- Полк**: A green button labeled "Поиск".
- Тип жилья**: A dropdown menu with options like "Квартира", "Дом", etc.
- Ценовой диапазон**: A dropdown menu set to "20 000 – 40 000 ₽".
- Особенности**: A dropdown menu with options like "Балкон", "Лифт", etc.
- Оплата**: A dropdown menu with options like "Наличные", "Банковской картой", etc.
- Расположение**: A dropdown menu with options like "Рядом с метро", "У центра", etc.
- Кол-во комнат**: A dropdown menu with options like "1", "2", "3", "Более".

On the right, two search results are displayed in cards:

- 27000 ₽ / месяц**
Екатеринбург, ул. Мира, 25
34 м², 1 комнат(-ы), 4-й этаж
Test description 2
- 25000 ₽ / месяц**
Санкт-Петербург, Пулковское шоссе, 71к3
29 м², 1 комнат(-ы), 4-й этаж
Test description 4

A large green button at the bottom left is labeled "Применить" (Apply).

Рисунок 8 – Отфильтрованные объекты недвижимости

Карточка недвижимости

Была добавлена страница для арендуемой недвижимости (см. рисунок 9).



Квартира 110000 ₽ / месяц

Адрес: Москва, Подсосенский переулок, 23с3

Характеристики

Площадь: 67 м²

Этаж: 2-й

Тип: Квартира

Комнат: 2

Расположение: Рядом с метро

Особенности: С балконом, с животными

Контакты владельца

Василий Петрович

Телефон: +7 900 555-44-33

Email: owner@example.com

[Написать владельцу](#)

[Описание](#)

Рисунок 9 – Страница недвижимости

Данные по недвижимости подтягиваются отдельным запросом, после чего специальная функция вставляет полученные данные в элементы html страницы (см. рисунок 10).

```

function fillPropertyContent(property) {
  const carousel = document.getElementById("carousellInner");
  const images = (property.images !== undefined && property.images.length !== 0) ? property.images : [DEFAULT_PICTURE_URL];

  images.forEach((img) => {
    imgDiv = document.createElement("div");
    imgDiv.className = "carousel-item active";
    imgDiv.innerHTML = ``;
    carousel.append(imgDiv);
  });

  const title = document.getElementById("propertyTitle");
  title.textContent = property.title;

  const price = document.getElementById("propertyPrice");
  price.textContent = `${property.price} ₽ / `;
  let frequency = "месяц";

  if (property.is_daily_payment)
    frequency = "день";

  price.textContent += frequency;

  const address = document.getElementById("propertyAddress");
  address.innerHTML = `<b>Адрес:</b> ${property.address}`;

  const specs = document.getElementById("propertySpecs");
  let floor = (property.floor === undefined) ? property.floor + "-й" : "-";

  specs.innerHTML =
    `<h5 class="mb-3">Характеристики</h5>
    <p><b>Площадь:</b> ${property.area} м2</p>
    <p><b>Этаж:</b> ${floor}</p>
    <p><b>Тип:</b> ${typeMapper.get(property.type)}</p>
    <p><b>Комнат:</b> ${property.rooms}</p>
    <p><b>Расположение:</b> Рядом с метро</p>
    <p><b>Особенности:</b> С балконом, с животными</p>
  `;

  const description = document.getElementById("propertyDescription");
  description.innerHTML =
    `<h5 class="mb-3">Описание</h5>
    ${property.description}
  `;
}

```

Рисунок 10 – Заполнение страницы недвижимости

Выводы

Выполнение данной лабораторной работы стало отличным упражнением во взаимодействии с внешним API со стороны фронтенда. Конечно, сейчас мало кто разрабатывает сайты, используя многостраничный подход с чистым js, однако на мой взгляд, опыт использования классического js + html улучшает понимание современных фреймворков и библиотек, асинхронного взаимодействия и того, как в целом формируется динамический контент на странице сайта.