

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

**Лабораторная работа 2: Лабораторная работа 2:
взаимодействие с внешним API**

Выполнил:

Оспельников Алексей

Группа К3440

**Проверил:
Добряков Д. И.**

Санкт-Петербург

2025 г.

Задача

Варианты остаются прежними. Теперь 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".

Например, для приложения для просмотра прогнозов погоды задание выглядит следующим образом:

Реализовать получение погоды (прогноз на ближайшие 7 дней) из открытого API OpenWeatherMap, в зависимости от геолокации пользователя. Реализовать вывод полученного прогноза в виде 7 карточек в три ряда (первый ряд - крупная карточка, второй ряд - три карточки в меньшем размере, третий ряд - четыре карточки в маленьком размере).

Вариант: Сервис для аренды недвижимости

- Вход
- Регистрация
- Личный кабинет пользователя (список арендованных и арендующихся объектов)
- Поиск недвижимости с фильтрацией по типу, цене, расположению
- Страница объекта недвижимости с фото, описанием и условиями аренды
- История сообщений и сделок пользователя

Каждый из сайтов обязательно должен включать в себя следующие страницы:

- Вход
- Регистрация
- Личный кабинет пользователя
- Страница для поиска с возможностью фильтрации

Ход работы

Первым делом, был создан mock-сервер с авторизацией и mock-db. Код сервера выглядит следующим образом:

```
const jsonServer = require('json-server');

const server = jsonServer.create();

const router = jsonServer.router('db.json');

const middlewares = jsonServer.defaults();

server.use(middlewares);

server.use(jsonServer.bodyParser);

server.post('/login', (req, res) => {

  const { username, password } = req.body;

  if (username === 'user' && password === 'pass') {

    return res.json({ token: 'fake-jwt-token' });

  }

  res.status(401).json({ error: 'Invalid credentials' });

});

server.use((req, res, next) => {

  if (req.method === 'GET' || req.url.startsWith('/login')) {

    return next();

  }

  const auth = req.headers.authorization;

  if (!auth || auth !== 'Bearer fake-jwt-token') {

    return res.status(401).json({ error: 'Unauthorized' });

  }

  next();
});
```

```
) ;

server.use(router);

server.listen(3001, () => {
  console.log('JSON Server is running on port 3001');
}) ;
```

Mock-db выглядит вот так:

```
{
  "users": [
    {
      "id": 1,
      "username": "user",
      "email": "user@example.com"
    }
  ],
  "properties": [
    {
      "id": 1,
      "title": "Уютная квартира в центре",
      "location": "Москва",
      "price": 5000,
      "type": "apartment",
      "images": ["image1.jpg", "image2.jpg"],
      "description": "Квартира с видом на парк",
      "ownerId": 1
    }
  ]
}
```

```
],  
  
"rentals": [  
  
  {  
  
    "id": 1,  
  
    "userId": 1,  
  
    "propertyId": 1,  
  
    "startDate": "2025-01-10",  
  
    "endDate": "2025-01-20",  
  
    "status": "active"  
  
  }  
  
]  
}
```

Вывод

В ходе работы удалось создать mock-сервер и mock-db для сервиса по аренде недвижимости