**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# **Дисциплина:** Фронт-энд разработка

Отчет

## Лабораторная работа №2

Выполнил:

Шалунов Андрей

Группа К3440

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

**Задача**

Нужно привязать то, что сделано 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 "Иммитация работы с API".

**Ход работы**

Вместо простого JSON сервера в качестве внешнего API я использовал готовый микросервисный backend, разработанный в прошлом семестре.

Структура:

1. api-gateway – единая точка входа для фронтенда
2. auth-service – авторизация и выдача токенов
3. user-service – управление пользователями и профилями
4. property-service – объекты недвижимости
5. photo-service – фотографии к объектам
6. booking-service – бронирования и сделки
7. message-service – сообщения между пользователями

**Страница регистрации**

Страница регистрации предназначена для создания нового аккаунта в системе. Обработчик в register-page.js проверяет, что обязательные поля форсы заполнены, и отправляет данные на эндпоинт POST /auth/register внешнего API. В случае успешного ответа показывается модальное окно с текстом «Аккаунт создан» и кнопкой перехода на страницу входа.

**Страница входа**

Страница входа содержит форму с полями email и пароль. После нажатия кнопки Войти скрипт signin-page.js отправляет введенные данные на внешний API (POST /auth/login) через общий модуль api.js. При успешной авторизации от бэкенда приходит токен, который сохраняется в localStorage с помощью функций из session.js, дальше по токену фронтенд может обращаться к защищённым эндпоинтам.

**Страница профиля**

При загрузке скрипт profile-page.js проверяет наличие токена, а затем через API запрашивает текущий профиль (GET /users/me) и подставляет имя, email и телефон. Далее выполняется запрос к сервису бронирований (GET /bookings), на основе которого формируются два списка: объекты, которые пользователь арендует, и объекты, которые он сдает, каждый выводится в виде карточек. Параллельно формируется таблица «История сделок», где для каждой записи показывается тип операции, объект, цена, адрес и статус. Отдельный блок «История сообщений» заполняется данными, полученными через fetchMyChats() из модуля messages.js.

**Страница редактирования профиля**

При открытии скрипт profile-edit-page.js проверяет авторизацию и через fetchMyProfile() запрашивает текущие данные профиля с API, затем заполняет ими поля формы. Пользователь может отредактировать информацию и нажать кнопку Сохранить, в этот момент выполняется запрос PATCH /users/:id через общий модуль api.js. Если обновление прошло успешно, отображается модальное окно saveModal с текстом Профиль обновлен и кнопкой Вернуться в профиль, которая ведет обратно в личный кабинет.

**Страница поиска недвижимости**

В верхней части находятся фильтры: тип недвижимости, город, а также минимальная и максимальная цена. Скрипт search-page.js по кнопке Применить фильтр собирает значения формы и вызывает fetchProperties(), который формирует query-параметры и отправляет запрос GET /properties на внешний API. Для каждого полученного объекта дополнительно запрашиваются фотографии через GET /photos/by-property/:propertyId, чтобы отобразить актуальное изображение в карточке.

**Страница объекта недвижимости**

Страница property.html показывает детальную информацию по конкретному объекту недвижимости. Отдельным запросом GET /photos/by-property/:propertyId загружается список фотографий, которые выводятся в виде галереи: крупное основное фото и ряд миниатюр, между которыми можно переключаться стрелками и кликом. Ниже находится форма Бронирование: пользователь выбирает даты заезда и выезда,

фронтенд рассчитывает стоимость и отправляет запрос POST /bookings, а затем обновляет статус объекта через PUT /properties/:id при успешном бронировании.

**Вывод**

В результате лабораторной работы верстка сервиса аренды недвижимости из ЛР1 была полностью привязана к внешнему микросервисному API с авторизацией по токену. Все основные страницы (вход, регистрация, профиль, поиск, объекты и бронирования) теперь обмениваются данными с бэкендом через HTTP запросы и работают с реальными данными.