

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)**

**Факультет прикладной информатики
Образовательная программа: Интеллектуальные системы в гуманитарной
сфере
Направление подготовки: 45.03.03 Интеллектуальные системы в гуманитарной
сфере**

**О Т Ч Е Т
по выполнению 2 лабораторной работы по предмету “Фронтенд
разработка”**

Обучающиеся: Иржанова Юлия Ирназаровна

**Санкт-Петербург,
2025**

Тема: Взаимодействие с внешним API (JSON-server)

Вариант: Музыкальный каталог оценки альбомов .

Цель работы

- Научиться работать с внешним API с помощью fetch/axios.
- Реализовать моковое REST-API через JSON-server и json-server-auth для авторизации.
- Перевести функционал ЛР1 (каталог, детали альбома, авторизация, профиль) на получение данных из API.

Структура проекта

The screenshot shows a file explorer window with the following structure:

- LAB-1 (selected folder)
- node_modules
- public
 - css
 - styles.css
 - data
 - albums.json
 - img
 - js
 - album.js
 - auth.js
 - catalog.js
 - profile.js
 - storage.js
 - index.html
 - login.html
 - profile.html
 - register.html
- db.json
- package-lock.json
- package.json
- ReadMe.md
- server.js

Взаимодействие с внешним API

JSON-server и структура API

В db.json описаны сущности: альбомы, пользователи и связанные с ними данные (например, избранные/отзывы).

JSON-server поднимает REST-API, которое отдаёт список альбомов (/albums), конкретный альбом (/albums/:id), а также эндпоинты авторизации и работы с пользователями через json-server-auth.

server.js конфигурирует запуск сервера, подключает JSON-server, описывает middleware и маршруты, а также интегрирует авторизацию (регистрация и логин через API).

```
JS server.js > ...
1  const jsonServer = require('json-server');
2  const auth = require('json-server-auth');
3  const path = require('path');
4
5  const app = jsonServer.create();
6  const router = jsonServer.router('db.json');
7
8  app.db = router.db;
9
10 // статика из ./public
11 app.use(jsonServer.defaults({
12   static: path.join(__dirname, 'public')
13 }));
14
15 app.use(auth);
16 app.use(router);
17
18 app.listen(3000, () => {
19   console.log('JSON Server with auth is running on http://localhost:3000');
20 });
21
```

Клиентские запросы (public/js)

Файлы в public/js реализуют клиентскую логику:

- [catalog.js](#)

Загружает список альбомов с API (GET /albums), применяет фильтрацию по жанру, году, рейтингу и рендерит карточки в каталоге.

- **album.js**

Получает ID альбома из URL, запрашивает данные конкретного альбома (GET /albums/:id), подставляет название, обложку, треки и рейтинг на страницу album-detail.html.

- **auth.js**

Отвечает за регистрацию и авторизацию через API: отправляет POST-запросы на эндпоинты логина/регистрации, обрабатывает ответы сервера и сохраняет токен/пользователя.

- **profile.js**

Запрашивает данные текущего пользователя и связанные сущности (например, список выбранных альбомов) с помощью токена; отображает их на profile.html.

- **storage.js**

Инкапсулирует работу с localStorage для сохранения токена, ID пользователя и вспомогательных данных, чтобы не дублировать эту логику.

Запросы к защищённым эндпоинтам выполняются с заголовком Authorization: Bearer <token>, который формируется на основе сохранённого токена в storage.js.

Привязка ЛР1 к API

Интерфейс, свёрстанный в ЛР1, полностью сохранён в папке public: структура страниц, Bootstrap-оформление, формы входа/регистрации, каталог и личный кабинет.



Основное изменение ЛР2 — данные теперь берутся не из “захардкоженного” массива в JS, а из внешнего API JSON-server, с которым приложение взаимодействует через JS-файлы в public/js.

Получение каталога альбомов

Каталог (index.html, catalog.html):

- при загрузке страницы catalog.js отправляет запрос к /albums;
- полученные данные преобразуются в карточки;
- при изменении фильтров выполняется повторная фильтрация либо новый запрос с параметрами, после чего DOM перерисовывается.

Таким образом, список альбомов всегда соответствует актуальному состоянию данных в db.json.

```
1 √ {
2 √   "users": [
3 √     {
4       "email": "aaa@mail.ru",
5       "password": "$2a$10$00pZj8u2MsaJVUG.jXzjLOIFC0mynwByCadB55hkFuqkbnlac2h7G",
6       "name": "yulya",
7       "id": 1
8     },
9     {
0       "email": "dfjldskj@mail.ru",
1       "password": "$2a$10$bJ4Rqn.5r5.Mzo77osPbe.ZLgTfcpnMsJMMhbQ0Dvvcdg4TjmLkX2",
2       "name": "fjfkd",
3       "id": 2
4     },
5     {
6       "email": "123@gmail.ru",
7       "password": "$2a$10$ZakxcqXPKomIRUPjln7wsuZxKZ5J4TEdI5tBwORWbXNy4TKjTJAka",
8       "name": "123",
9       "id": 3
0     }
1   ],
2   "albums": [
3     {
4       "id": 1,
5       "albumTitle": "Man's Best Friend",
6       "artist": "Sabrina Carpenter",
7       "genre": "Pop",
8       "year": 2025,
9       "rating": 7.8,
0       "description": "Седьмой студийный альбом Сабрины Карпентер с элементами диско-попа, фанка и
1       "coverUrl": "img/sabrina.jpeg",
2       "tracks": [
3         "Manchild",
4         "Tears",
5         "My Man on Willpower",
6         "Sugar Talking",
7         "We Almost Broke Up Again Last Night",
8         "Coincidence",
9         "Sharpest Tool",
0         "Slim Pickins",
1         "Juno",
2         "A Thousand Times",
3         "Taste of You",
4         "Goodbye"
5       ]
6     }
7   ]
8 }
```

Детали альбома

Страница album-detail.html получает ID альбома из параметров URL и с помощью album.js запрашивает у API подробные данные, включая описание и трек-лист.

Скрипт формирует разметку: подставляет заголовок, обложку, список треков и дополнительные поля (жанр, год).

Авторизация через API

- auth.js реализует работу с эндпоинтами авторизации JSON-server:

- отправка формы регистрации (POST на /register или аналогичный маршрут, создающий пользователя в db.json);
- отправка формы входа (POST на /login), получение токена и информации о пользователе;
- сохранение токена и id пользователя в storage.js и перенаправление на профиль/каталог.

При наличии токена профиль и другие защищённые экраны могут делать запросы от имени конкретного пользователя.

Личный кабинет и работа с пользователем

profile.js использует сохранённый токен, чтобы запросить данные текущего пользователя (GET /users/:id или соответствующий эндпоинт).

На основе полученных данных динамически заполняется страница profile.html: имя, email, список избранных альбомов или действий, чтобы профиль зависел от ответа API, а не от статичных данных.

Заключение

Во 2-й лабораторной работе статический музыкальный каталог из ЛР1 был интегрирован с внешним API, реализованным на JSON-server, а авторизация и работа с пользователем переведены на обмен данными с сервером.

Теперь все ключевые страницы (каталог, страница альбома, вход, регистрация, профиль) получают данные из API и обновляют интерфейс динамически, что соответствует имитации реального бэкенда.