**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Фронт-энд разработка

# Отчет

## Лабораторная работа №2

Выполнила:

Красюк Карина

K3441

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

**СОДЕРЖАНИЕ**

**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 "Иммитация работы с API".

Вариант “Сайт для поиска работы”.

**Ход работы**

**1 Краткое резюме выполненного**

Интегрировано приложение JobFinder с моковым API на базе JSON-server.

Реализованы:

- Авторизация через API (вход/регистрация)
- Загрузка вакансий с сервера
- CRUD для вакансий, резюме и откликов
- Middleware для обработки авторизации
- Работа с токенами через localStorage

**2 Список реализованных функций с примерами кода**

**2.1 Настройка JSON-server с middleware**

Создан server-middleware.js для обработки авторизации:

```
module.exports = (req, res, next) => {
  // CORS заголовки
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, PATCH, OPTIONS');
  res.header('Access-Control-Allow-Headers', 'Content-Type, Authorization');

  // Эндпоинт для авторизации
  if (req.path === '/login' && req.method === 'POST') {
    const { email, password, role } = req.body;
    const db = require('./db.json');
```

```
        const user = db.users.find(
          u => u.email === email && u.password === password &&
u.role === role
        );

        if (user) {
          const token = `token_${user.id}_${Date.now()}`;
          return res.json({
            accessToken: token,
            user: { id: user.id, email: user.email, role:
user.role, ...user }
          });
        } else {
          return res.status(401).json({ error: 'Неверный email
или пароль' });
        }
      }
      // ... остальной код
    };
```

### 2.2 API модуль для работы с сервером

Создан js/api.js с функциями для работы с API:

```
const API_BASE_URL = 'http://localhost:3000';

// Базовая функция для запросов
async function apiRequest(endpoint, options = {}) {
  const url = `${API_BASE_URL}${endpoint}`;
  const token = getToken();

  const config = {
    ...options,
    headers: {
      'Content-Type': 'application/json',
      ...(token && { Authorization: `Bearer ${token}` }),
      ...options.headers,
```

```
    },
  };

  try {
    const response = await fetch(url, config);
    const data = await response.json();
    if (!response.ok) {
      throw new Error(data.error || `Ошибка:
${response.status}`);
    }
    return data;
  } catch (error) {
    console.error('API Error:', error);
    throw error;
  }
}

// API для авторизации
export const authAPI = {
  async login(email, password, role) {
    const data = await apiRequest('/login', {
      method: 'POST',
      body: JSON.stringify({ email, password, role }),
    });

    if (data.accessToken) {
      setToken(data.accessToken);
      setUser(data.user);
    }
    return data;
  },

  async register(userData) {
    const newUser = await apiRequest('/users', {
```

```
            method: 'POST',
            body: JSON.stringify({
              email: userData.email,
              password: userData.password,
              role: userData.role,
              ...(userData.role === 'candidate'
                ? { name: userData.name, position:
userData.position, city: userData.city }
                : { companyName: userData.companyName, industry:
userData.industry, city: userData.city }
              )
            }),
          });

          if (userData.role === 'candidate') {
            await apiRequest('/resumes', {
              method: 'POST',
              body: JSON.stringify({
                userId: newUser.id,
                position: userData.position || '',
                salary: 0,
                city: userData.city || '',
                // ... остальные поля
              }),
            });
          }

          return { success: true, user: newUser };
        },

        logout() {
          removeToken();
          removeUser();
        },
```

```
  getCurrentUser() {
    return getUser();
  },

  isAuthenticated() {
    return !!getToken();
  },
};
```

### 2.3 Загрузка вакансий на главной странице

Реализована функция загрузки вакансий с сервера:

```
async function loadJobs() {
  const jobsList = document.getElementById('jobsList');
  if (!jobsList) return;

  try {
    const jobs = await jobsAPI.getAll();
    jobsList.innerHTML = '';

    jobs.forEach((job) => {
      const jobItem = document.createElement('a');
      jobItem.href = `job-detail.html?id=${job.id}`;
      jobItem.className = 'list-group-item list-group-item-
action job-item';
      jobItem.setAttribute('data-industry', job.industry);
      jobItem.setAttribute('data-experience',
job.experience);
      jobItem.setAttribute('data-salary', job.salary);

      jobItem.innerHTML = `
        <div class="d-flex w-100 justify-content-between">
          <div>
            <h5 class="mb-1">${job.title}</h5>
```

```
                <p class="mb-1 text-muted">${job.company} •
${job.location} • ${job.employmentType}</p>
                <span class="badge bg-primary me-
1">${industryLabels[job.industry]}</span>
                <span class="badge bg-secondary me-
1">${expLabels[job.experience]}</span>
              </div>
              <div class="text-end">
                <div class="h5 mb-1">от
${job.salary.toLocaleString('ru-RU')} ₽</div>
                <small class="text-muted">Опубликовано:
${dateText}</small>
              </div>
            </div>
          `;
          jobsList.appendChild(jobItem);
        });

        applyFilters();
      } catch (error) {
        console.error('Ошибка загрузки вакансий:', error);
        showMessage('Ошибка загрузки вакансий', 'error');
      }
    }
```

### 2.4 Обработка форм входа и регистрации

Реализована обработка форм через API:

```
    function initLoginForms() {
      const candidateForms =
document.querySelectorAll('form[data-role="candidate-login"]');
      candidateForms.forEach((form) => {
        form.addEventListener('submit', async (e) => {
          e.preventDefault();
          if (!form.checkValidity()) {
            e.stopPropagation();
```

```
            form.classList.add('was-validated');
            return;
          }

          const email =
form.querySelector('input[type="email"]').value;
          const password =
form.querySelector('input[type="password"]').value;

          try {
            const data = await authAPI.login(email, password,
'candidate');
            showMessage('Успешный вход!');
            setTimeout(() => {
              window.location.href = 'user-dashboard.html';
            }, 500);
          } catch (error) {
            showMessage(error.message || 'Ошибка входа',
'error');
          }
        });
      });
    }
```

### 2.5 Загрузка данных в личном кабинете соискателя

Реализована загрузка резюме и откликов:

```
async function loadUserDashboard() {
  const user = authAPI.getCurrentUser();
  if (!user || user.role !== 'candidate') {
    window.location.href = 'login.html';
    return;
  }

  try {
    const resume = await resumesAPI.getByUserId(user.id);
```

```
        if (resume) {
          // Заполняем резюме
          const positionEl = document.querySelector('.resume-
block-title + div');
          if (positionEl) positionEl.textContent =
resume.position || 'Не указано';

          // Загружаем навыки
          const skillsContainer =
document.querySelector('.badge.bg-primary')?.parentElement;
          if (skillsContainer && resume.skills) {
            skillsContainer.innerHTML = resume.skills.map(skill
=>
              `<span class="badge bg-primary me-
1">${skill}</span>`
            ).join('');
          }
        }

        // Загружаем отклики
        const applications = await
applicationsAPI.getByUserId(user.id);
        const applicationsList = document.querySelector('.list-
group.list-group-flush');
        if (applicationsList && applications.length > 0) {
          applicationsList.innerHTML = '';
          for (const app of applications) {
            const job = await jobsAPI.getById(app.jobId);
            // ... создание элементов списка откликов
          }
        }
      } catch (error) {
        console.error('Ошибка загрузки данных:', error);
      }
```

```
}
```

### 2.6 Создание вакансий в кабинете работодателя

Реализовано создание вакансий через API:

```
const newVacancyForm =
document.querySelector('#newVacancyModal form');
if (newVacancyForm) {
  newVacancyForm.addEventListener('submit', async (e) => {
    e.preventDefault();
    if (!newVacancyForm.checkValidity()) {
      e.stopPropagation();
      newVacancyForm.classList.add('was-validated');
      return;
    }

    const inputs = newVacancyForm.querySelectorAll('input,
select, textarea');
    const title = inputs[0].value;
    const salary = Number(inputs[1].value) || 0;
    const location = inputs[2].value;
    const industry = inputs[3].value.toLowerCase();
    const experience = inputs[4].value.toLowerCase();
    const description = inputs[5].value;

    try {
      await jobsAPI.create({
        title,
        company: user.companyName,
        location,
        salary,
        industry,
        experience,
        employmentType: 'Полная занятость',
        workFormat: 'Офис',
        description,
```

```
            responsibilities: [],
            requirements: [],
            conditions: [],
          });
          showMessage('Вакансия успешно создана!');
          const modal =
bootstrap.Modal.getInstance(document.getElementById('newVacancyMod
al'));
          if (modal) modal.hide();
          newVacancyForm.reset();
          setTimeout(() => location.reload(), 1000);
        } catch (error) {
          showMessage(error.message || 'Ошибка создания
вакансии', 'error');
        }
      });
    }
```

### 2.7 Отклики на вакансии

Реализована отправка откликов:

```
    // В функции loadJobDetail()
    const     applyBtn     =     document.querySelector('[data-bs-
target="#applyModal"]');
    if (applyBtn) {
      applyBtn.addEventListener('click', () => {
        const user = authAPI.getCurrentUser();
        if (user && user.role === 'candidate') {
          // Показываем кнопку подтверждения
          applyModalBody.innerHTML = `
            <p  class="mb-3">Вы  хотите  откликнуться  на  вакансию
"${job.title}"?</p>
            <button id="confirmApplyBtn" class="btn btn-primary w-
100">Откликнуться</button>
          `;
```

```
            const confirmBtn =
document.getElementById('confirmApplyBtn');
            if (confirmBtn) {
              confirmBtn.addEventListener('click', async () => {
                try {
                  await applicationsAPI.create(jobId);
                  showMessage('Отклик успешно отправлен!');
                  const modal =
bootstrap.Modal.getInstance(document.getElementById('applyModal'))
;
                  if (modal) modal.hide();
                } catch (error) {
                  showMessage(error.message || 'Ошибка отправки
отклика', 'error');
                }
              });
            }
          }
        });
      }
```

**3 Демонстрация проделанной работы в виде визуализаций**

Рисунок 1: Network в DevTools с запросами к API

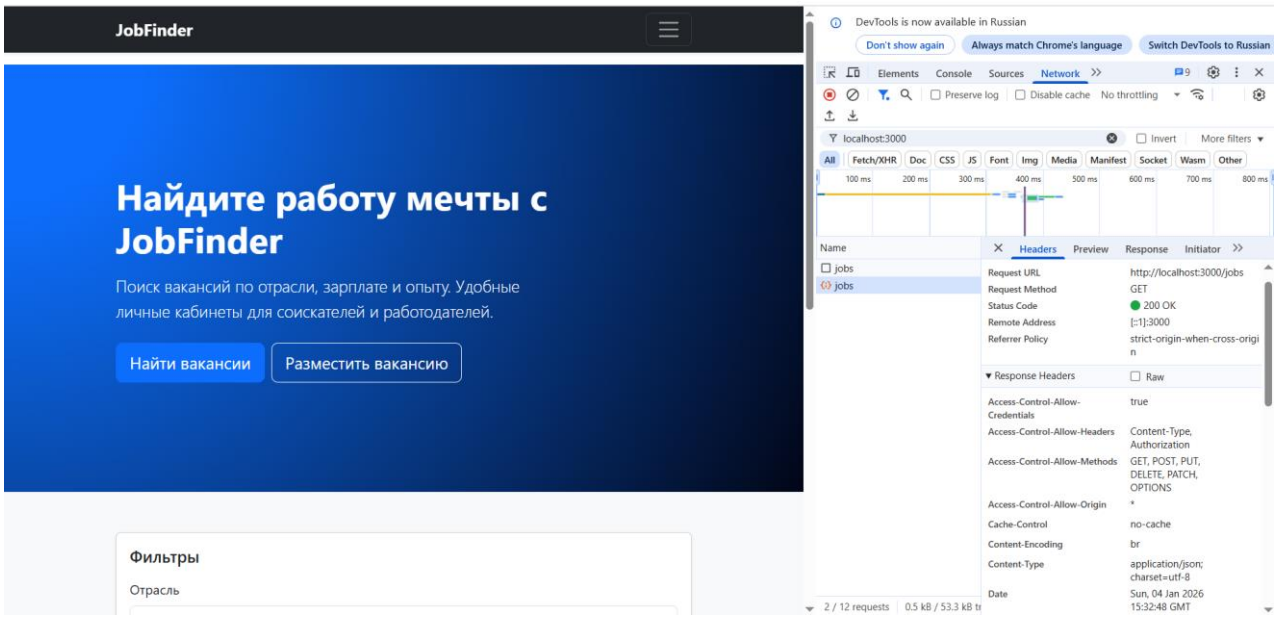
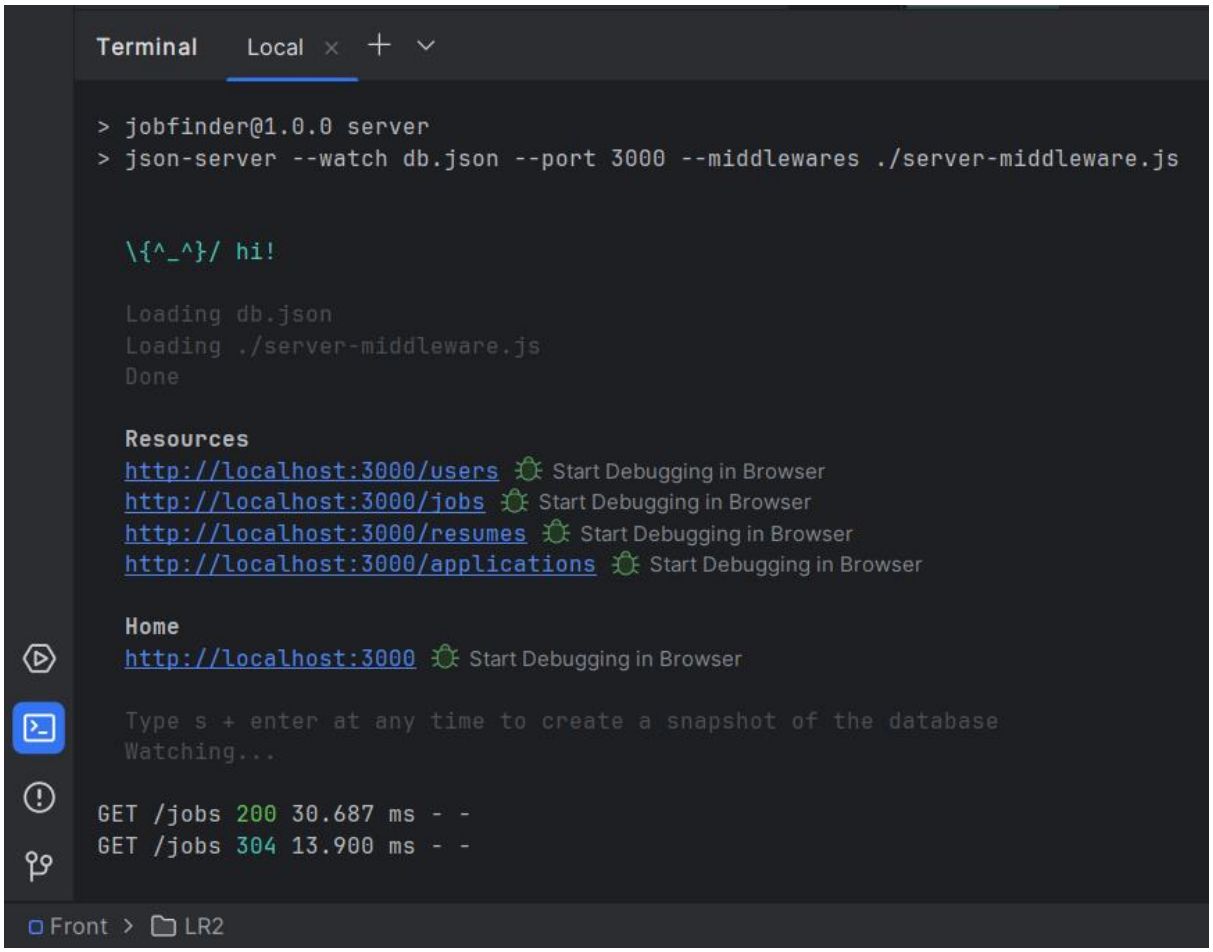


Рисунок 2: JSON-server в терминале



**Вывод**

Сайт готов к использованию. JSON-server работает как моковое API, все операции выполняются через HTTP-запросы. Код структурирован, ошибки обрабатываются, пользовательский опыт улучшен.

Текущая реализация соответствует требованиям задания и демонстрирует работу с внешним API средствами fetch.