САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа 2: Взаимодействие с внешним API

Выполнили:
Карнаухова Елизавета
Динкель Алиса
K3344

Проверил:
Добряков Д. И.

Санкт-Петербург

2024 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 "Иммитация работы с API".

Ход работы

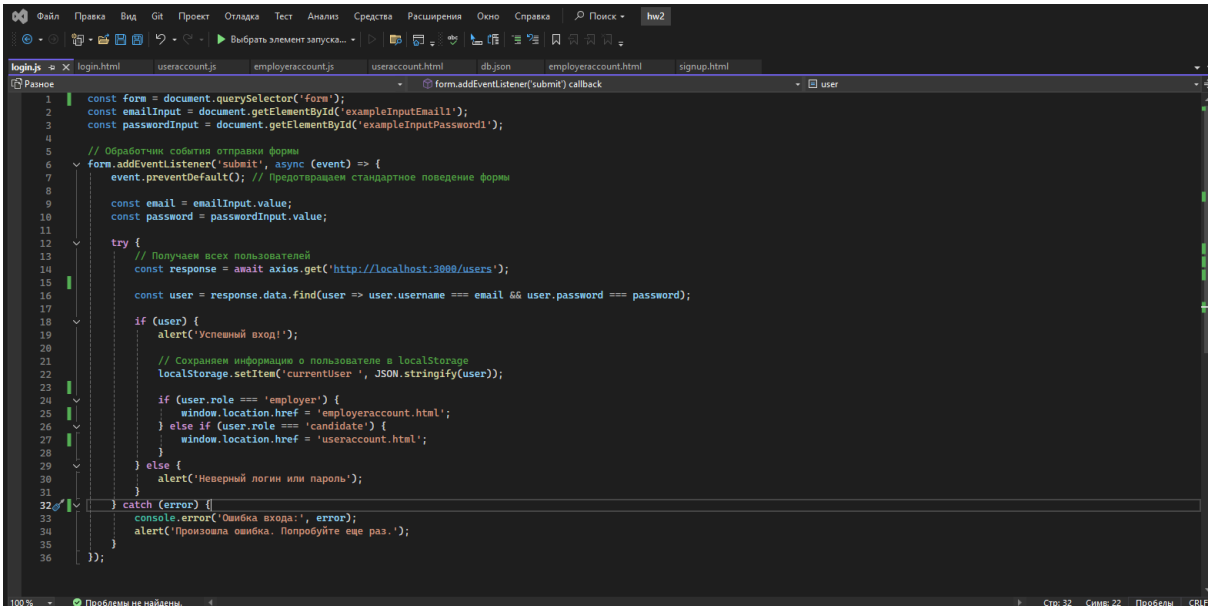
Создание json сервера

С помощью команды npm install json-server установили json сервер, и изменили файл db под наш сайт

Файл db.json

```
{
  "users": [
    {
      "id": "1",
      "username": "employer1@mail.ru",
      "password": "password123",
      "role": "employer"
    },
    {
      "id": "2",
      "username": "user1@mail.ru",
      "password": "password456",
      "role": "candidate"
    },
    {
      "id": "202a",
      "username": "employer2@mail.ru",
      "password": "123",
      "role": "employer"
    }
  ],
  "vacancies": [
    {
      "id": "bcec",
      "title": "Руболес",
      "description": "Рубить лес",
      "salary": "10.000",
      "experience": "Middle",
      "employerId": "202a"
    }
  ],
  "companies": [
    {
      "id": "1",
      "username": "employer2@mail.ru",
      "name": "Сосенки",
      "description": "Продажа сосен оптом, давно на рынке",
      "contacts": "sosniforsale@mail.ru"
```

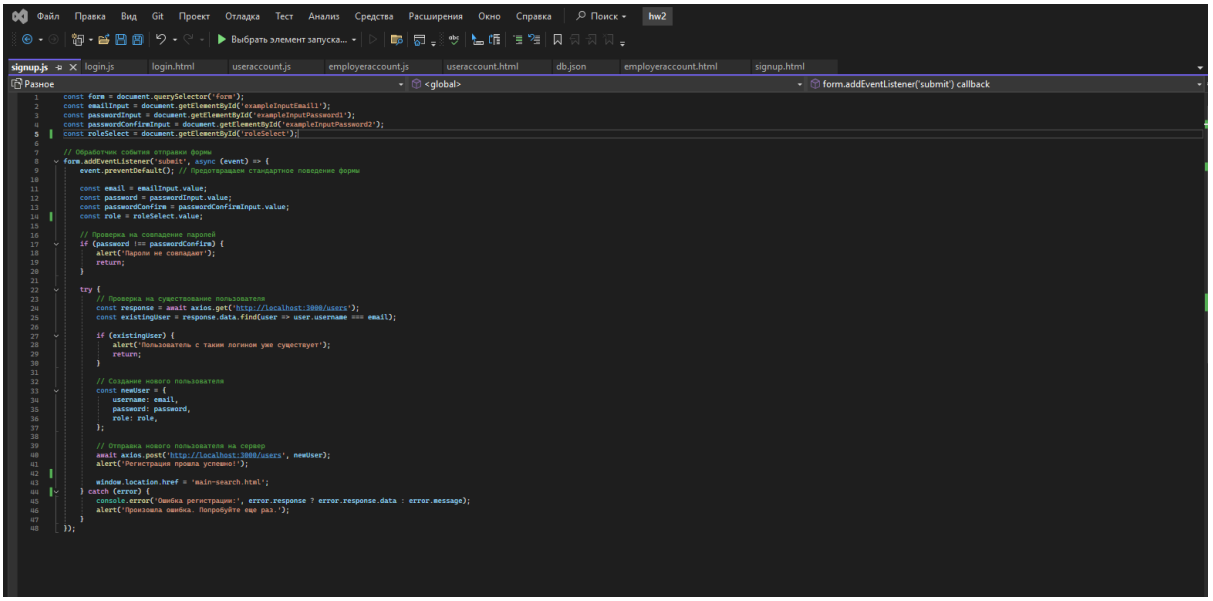
Создание login.js



```
const form = document.querySelector('form');
const emailInput = document.getElementById('exampleInputEmail1');
const passwordInput = document.getElementById('exampleInputPassword1');

// Обработчик события отправки формы
form.addEventListener('submit', async (event) => {
    event.preventDefault(); // Предотвращаем стандартное поведение формы

    const email = emailInput.value;
    const password = passwordInput.value;

    try {
        // Получаем всех пользователей
        const response = await axios.get('http://localhost:3000/users');

        const user = response.data.find(user => user.username === email && user.password === password);

        if (user) {
            alert('Успешный вход!');

            // Сохраняем информацию о пользователе в localStorage
            localStorage.setItem('currentUser ', JSON.stringify(user));

            if (user.role === 'employer') {
                window.location.href = 'employeraccount.html';
            } else if (user.role === 'candidate') {
                window.location.href = 'useraccount.html';
            }
        } else {
            alert('Неверный логин или пароль');
        }
    } catch (error) {
        console.error('Ошибка входа:', error);
        alert('Произошла ошибка. Попробуйте еще раз.');
    }
});
```

Напишем скрипт для обработки авторизации, и сохраним текущего пользователя в localStorage, также делаем редирект в зависимости от роли, либо работодатель, либо работник.

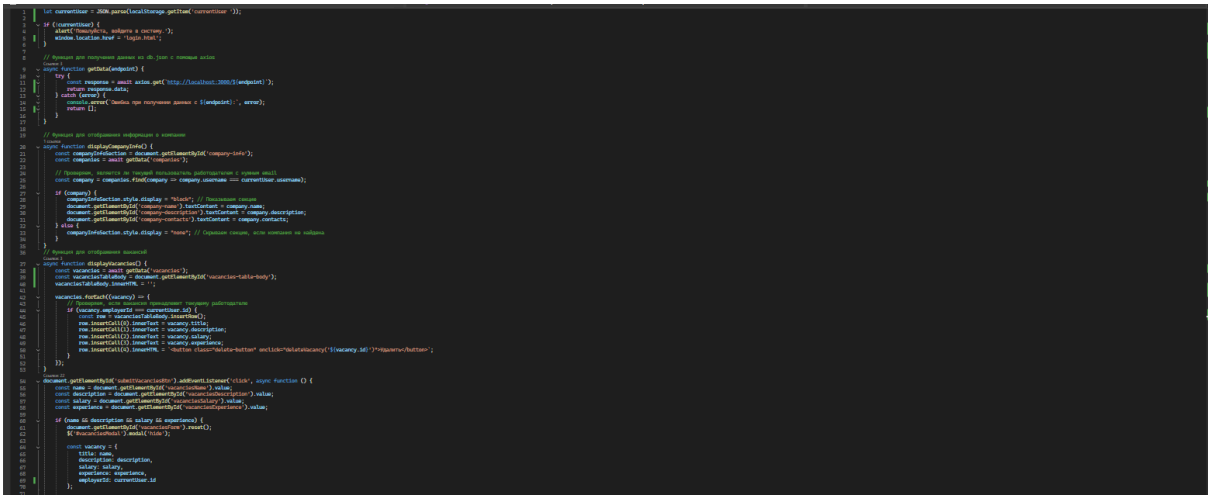
Создание signup.js



```
const form = document.querySelector('form');
const emailInput = document.getElementById('exampleInputEmail1');
const passwordInput = document.getElementById('exampleInputPassword1');
const passwordConfirmInput = document.getElementById('exampleInputPassword2');
const roleSelect = document.getElementById('roleSelect');

// Обработчик события отправки формы
form.addEventListener('submit', async (event) => {
    event.preventDefault(); // Предотвращаем стандартное поведение формы

    const email = emailInput.value;
    const password = passwordInput.value;
    const passwordConfirm = passwordConfirmInput.value;
    const role = roleSelect.value;

    // Проверка на совпадение паролей
    if (password !== passwordConfirm) {
        alert('Пароли не совпадают');
        return;
    }

    try {
        // Проверка на существование пользователя
        const response = await axios.get('http://localhost:3000/users');
        const existingUser = response.data.find(user => user.username === email);

        if (existingUser) {
            alert('Пользователь с таким логином уже существует');
            return;
        }

        // Создание нового пользователя
        const newUser = {
            username: email,
            password: password,
            role: role,
        };

        // Отправка нового пользователя на сервер
        await axios.post('http://localhost:3000/users', newUser);
        alert('Регистрация прошла успешно!');

        window.location.href = 'main-search.html';
    } catch (error) {
        console.error('Ошибка регистрации:', error.response ? error.response.data : error.message);
        alert('Произошла ошибка. Попробуйте еще раз.');
    }
});
```

Пропишем логику регистрации, есть две проверки, одна на совподение паролей, другая на регистрацию e-mail, также каждый новый пользователь будет записывать в db.json

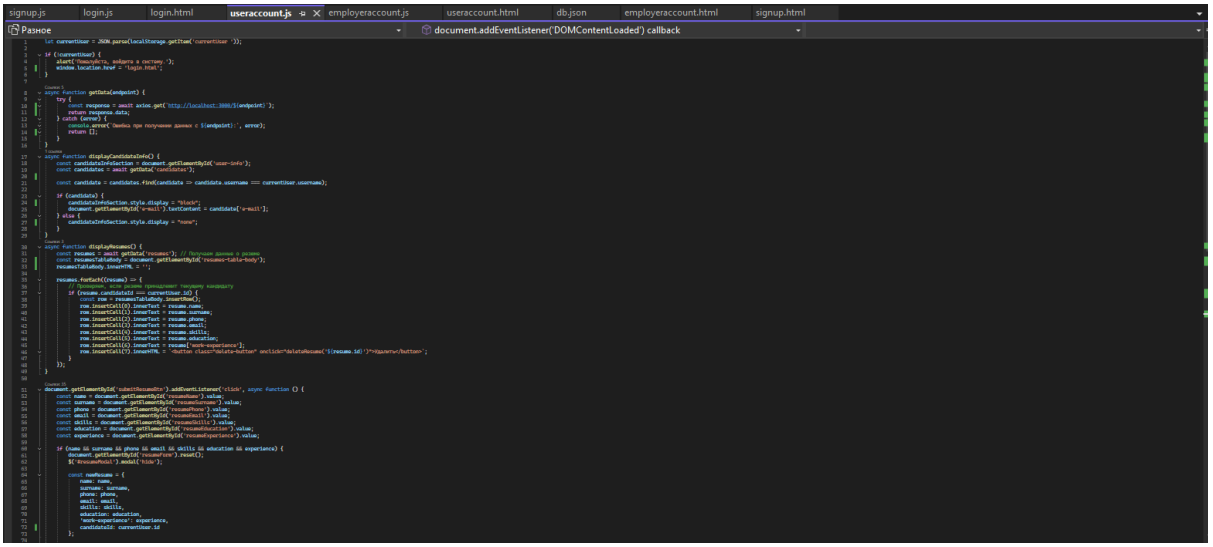
Создание emploeraccount.js



```
let currentUser = JSON.parse(localStorage.getItem('currentUser'));

if (currentUser) {
    alert('Пожалуйста, войдите в систему.');
    window.location.href = 'login.html';
}

// Функция для получения данных из db.json с помощью axios
async function getData(endpoint) {
    try {
        const response = await axios.get(`http://localhost:3000/${endpoint}`);
        return response.data;
    } catch (error) {
        console.error(`Ошибка при получении данных с ${endpoint}:`, error);
        return [];
    }
}

// Функция для отображения информации о компании
async function displayCompanyInfo() {
    const companyInfoSection = document.getElementById('company-info');
    const companies = await getData('companies');

    // Проверяем, является ли текущий пользователь работодателем с нужным email
    const company = companies.find(company => company.username === currentUser.username);

    if (company) {
        companyInfoSection.style.display = "block"; // Показываем секцию
        document.getElementById('company-name').textContent = company.name;
        document.getElementById('company-description').textContent = company.description;
        document.getElementById('company-contacts').textContent = company.contacts;
    } else {
        companyInfoSection.style.display = "none"; // Скрываем секцию, если компания не найдена
    }
}
// Функция для отображения вакансий
async function displayVacancies() {
    const vacancies = await getData('vacancies');
    const vacanciesTableBody = document.getElementById('vacancies-table-body');
    vacanciesTableBody.innerHTML = '';

    vacancies.forEach((vacancy) => {
        // Проверяем, если вакансия принадлежит текущему работодателю
        if (vacancy.employerId === currentUser.id) {
            const row = vacanciesTableBody.insertRow();
            row.insertCell(0).innerText = vacancy.title;
            row.insertCell(1).innerText = vacancy.description;
            row.insertCell(2).innerText = vacancy.salary;
            row.insertCell(3).innerText = vacancy.experience;
            row.insertCell(4).innerHTML = `<button class="delete-button" onclick="deleteVacancy('${vacancy.id}')">Удалить</button>`;
        }
    });
}
document.getElementById('submitVacancies0tr').addEventListener('click', async function () {
    const name = document.getElementById('vacanciesName').value;
    const description = document.getElementById('vacanciesDescription').value;
    const salary = document.getElementById('vacanciesSalary').value;
    const experience = document.getElementById('vacanciesExperience').value;

    if (name && description && salary && experience) {
        document.getElementById('vacanciesForm').reset();
        $('#vacanciesModal').modal('hide');

        const vacancy = {
            title: name,
            description: description,
            salary: salary,
            experience: experience,
            employerID: currentUser.id
        };
```

Проверка на соответствие пользователя, только тогда отображается под него личная информация, отображение откликов и вакансии работают по той же логике

Создание userraccount.js



Если пользователь есть бд, отображает личную информацию и резюме, также сохраняет его в бд

Вывод

В ходе работы была проведена реализация моковоего API средствами JSON-сервера и подключите к нему авторизацию, регистрацию а также обработку личного кабинета работодателя, и работника.