Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

У Т В Е Р Ж Д А Ю

**Зав. кафедрой ВММБ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Ю. Столбов

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

***ЗАДАНИЕ***

***НА КУРСОВУЮ РАБОТУ***

***по дисциплине***

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Ким Елизавета, Лопина Мария, Решетникова Елизавета; группа: ИСТ-19-1б

(фамилия, имя, отчество студента; группа)

1. **Тема курсовой работы**

Автоматизированная информационная система для взаимодействия Соискателя и Работодателя

15 апреля 23

2. **Срок сдачи студентом отчета:** «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

3. **Содержание отчета:**

Был проведен анализ рассматриваемой области. Проведен анализ средств реализации и выбраны

подходящие. Частично разработан обозначенный в ТЗ функционал с учётом созданной БД.

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**КУРСОВАЯ РАБОТА**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Автоматизированная информационная система для взаимодействия Соискателя и Работодателя»**

Выполнили:

студент гр. ИСТ-19-1Б

Ким Елизавета Алексеевна

Лопина Мария Сергеевна

Решетникова Елизавета Андреевна

(Ф.И.О.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись)*

Принял:

ст. преподаватель, Банников Р.Ю.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО руководителя)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь 2023**

Содержание.

[1. Постановка задачи 6](#_Toc132570196)

[Требования к функциям отчета системы: 7](#_Toc132570197)

[Этапы реализации 8](#_Toc132570198)

[2. Средства разработки 9](#_Toc132570199)

[СУБД «PostgreSQL» 9](#_Toc132570200)

[JS, React и Visual Studio Code 9](#_Toc132570201)

[jsPDF 10](#_Toc132570202)

[3. Структура БД. 11](#_Toc132570203)

[3.1. Предметная область 11](#_Toc132570204)

[3.2. Логическая и физическая модель 11](#_Toc132570205)

[3.3. Описание сущностей БД 12](#_Toc132570206)

[4. Программный продукт 14](#_Toc132570207)

[4.1. Авторизация 14](#_Toc132570208)

[4.2. Функционал ИС 15](#_Toc132570209)

[4.3. Отчет 16](#_Toc132570210)

[4.4. Реакция ИС на ошибочный ввод данных 17](#_Toc132570211)

[5. Выводы 19](#_Toc132570212)

[Список литературы 20](#_Toc132570213)

[Приложение 1 21](#_Toc132570214)

[Приложение 2 22](#_Toc132570215)

1. Постановка задачи

Автоматизированная информационная система предназначена для автоматизации поиска тем для научных работ и Соискателей для выполнения задач, поставленных Работодателями.

Цели создания системы:

* Ведение базы данных с доступными задачами для их выполнения.
* Ведение базы данных с резюме Соискателей.
* Управление процессами взаимодействия между Соискателем и Работодателем.

Возможности системы:

* Сбор в одной информационной системе задач, представленных организациями для взаимодействия с исполнителями.
* Создание ведение карточек-задач со стороны Работодателей с подробной информацией о поставленной задаче.
* Создание и ведение карточек-анкет со стороны соискателей о имеющихся навыках и опыте.

Требования к системе:

Система должна быть многопользовательской. Данные для разных пользователей должны разделяться. Соискатель имеет возможность просмотреть анкету-задачу любого заказчика, опубликованную на ресурсе. Соискатели не имеют возможности просматривать личные данные других Соискателей, за исключением тех, которые соискатель опубликовал для общего доступа (прикрепленное резюме).

Работодатель имеет возможность опубликовать анкету-задачу для привлечения Соискателя, а также просматривать личную информацию и резюме откликнувшегося на задачу Соискателя.

Доступ в систему должен осуществляться через веб-браузер.

Требования к функциям авторизации и аутентификации пользователей:

Система должна позволять:

* вести учет пользователей;
* вести учет ролей пользователей;
* вести авторизацию и аутентификацию пользователей;
* вести раздельный доступ к объектам системы на основе ролей доступа;
* восстановление пароля.

Функционал ролей пользователей:

Роли:

А) Администратор

Б) Соискатель

В) Работодатель

Функционал ролей:

Администратор:

* редактирование пользователей;
* просмотр списка пользователей;
* удаление пользователей;
* редактирование прав доступа для пользователя;
* блокировка пользователей.

Соискатель:

* редактирование личной информации профиля;
* размещение резюме;
* просмотр списка опубликованных задач;
* отклик на задачу.

Работодатель:

* редактирование публичной информации профиля;
* размещение задачи;
* просмотр резюме Соискателей;
* подтверждение отклика от Соискателя.

## Требования к функциям отчета системы:

* система должна выдавать результаты поиска в соответствии с параметрами, выбранными в фильтре;
* система должна выводить отчет по результатам поиска в формате .pdf;
* в отчете фиксируется первые 100 результатов.

Отчеты***,*** выводимые в системе:

1. Отчет, выводящий задачи, отсортированные по дате публикации;
2. Отчет, выводящий профили Соискателей, являющиеся студентами выше 3 курса;
3. Отчет, выводящий все оплачиваемые задачи, для которых требуется знание Python или Java;
4. Отчет, выводящий задачи, опубликованные конкретным Работодателем и содержащие в своём описании «информационные технологии»;
5. Отчет, выводящий все оплачиваемые задачи и отсортированные по убыванию стоимости работы;
6. Отчет, выводящий список задач, опубликованных конкретным Работодателем;
7. Отчет, выводящий список Соискателей с Электротехнического факультета и Факультета прикладной математики и механики;
8. Отчет, выводящий задачи содержащий в своем описании «Нейронные сети».

Этапы реализации

На основании сформулированного задания были выделены 5 основных этапов реализации системы «UpsilonConnect» и определены сроки начала и окончания выполнения каждого этапа. Также выбрана форма отчетности после каждого этапа в виде отчета в формате doc. Подробная информация о сроках отображена в таблице 1.

Таблица 1 – Сроки реализации.

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер** | **Этап** | **Срок выполнения** | **Форма отчетности** |
| 1 | Проектирование хранилища | 27.02.2023 | Отчет |
| 2 | Проектирование интерфейса | 06.03.2023 | Отчет |
| 3 | Разработка диаграммы классов | 13.03.2023 | Отчет |
| 4 | Программная реализация | 27.03.2023 | Отчет и демонстрация |
| 5 | Тестирование | 10.04.2023 | Отчет |

1. Средства разработки

В предыдущем пункте были выделены основные этапы реализации разрабатываемого программного продукта. Для работы с базой данных была выбрана СУБД «PostgreSQL» и для написания кода – язык программирования JS совместно с фреймворком React в среде разработки Visual Studio Code. Библиотека для автоматизации формирования отчета использовалась jsPDF.

СУБД «PostgreSQL»

Перед тем как создать базу данных, необходимо определиться с типом системы управления базами данных (СУБД) [1].

Таблица 2 Анализ СУБД

|  |  |  |
| --- | --- | --- |
| **Тип СУБД** | **Когда выбирать** | **Примеры популярных СУБД** |
| Реляционные | Нужна транзакционность; высокая нормализация; большая доля операций на вставку | Oracle, MySQL, Microsoft SQL Server, PostgreSQL |
| Ключ-значение | Задачи кэширования и брокеры сообщений, для баз, где нужно хранить простые структуры данных и иметь быстрый доступ к ним | Redis, Memcached |
| Документные | Для хранения объектов в одной сущности, но с разной структурой; хранение структур на основе JSON | CouchDB, MongoDB, Amazon DocumentDB |
| Графовые | Задачи подобные социальным сетям; системы оценок и рекомендаций | Neo4j, Amazon Neptune, InfiniteGraph, InfoGrid |
| Колоночные | Хранилища данных; выборки со сложными аналитическими вычислениями; количество строк в таблице превышает сотни миллионов | Vertica, ClickHouse, Google BigTable, Sybase \ SAP IQ, InfoBright, Cassandra |

На этапе разработки логической модели в рамках работы над данной информационной системой стала понятно, что требуется хранить значительное количество сущностей (таблиц), с различными типами связей. В этом случае следует выбрать реляционную СУБД. Также необходимо отметить, что СУБД ключ-значение рассматривалось в виду простоты структуры базы данных учебного проекта. Но в будущем планируется увеличение сущностей, хранимых в БД, и по мере увеличения сложности запросов, выбор «ключ-значение» окажется не лучшим вариантом.

По итогу анализа была выбрана реляционная СУБД. PostgreSQL был выбран за безопасность баз данных. Он предоставляет доступ к объектам и данным базы данных через ROLES и PRIVILEGES с помощью команд GRANT. Возможности Postgres-а также имеет возможность создавать вложенные подзапросы, поддерживает триггеры, а также предоставляет одновременный доступа к БД с возможностью работы с данными для большого числа пользователей. Последний пункт был в рамках командной работы стал актуален.

JS, React и Visual Studio Code

Веб-приложение типа одностраничного приложения (SPA или Single Page Application)

было выбрана для реализации поставленных задач. SPA [2] – это веб-сайт, использующий единственный HTML документ как оболочку для всех страниц и организующий взаимодействие с пользователем через динамически подгружаемые данные, его компоненты модули и JavaScript и CSS-файлы загружаются один раз на одной странице. Его преимущества перед обычным сайтом \_ это отсутствие перегрузки, повторной отрисовки страницы, хорошее юзабиолити, но стоит учитывать, что при этом увеличивается нагрузка на браузер. Для разработки SPA необходим язык программирования JavaScript.

Фреймворк React был выбран [3, 4] за быстроту рендеринга, реактивность, компонентный подход и легковесный Virtual DOM.

Visual Studio Code (или VS Code) является редактором кода с функциями IDE, полностью бесплатный для личного и коммерческого использования, кроссплатформенный (macOS, Linux, Windows) и поддерживает необходимый в рамках проекта JavaScript и имеет поддержку контроля версий.

jsPDF

При реализации функции генерации отчета в веб-приложении использовалась библиотека jsPDF. Выбор обусловлен тем, что эта библиотека с открытым исходным кодом позволяет формировать отчеты на стороне клиента в формате .pdf. По jsPDF в открытом доступе расположено достаточное количество документации [5] и уроков, что позволило с легкостью интегрировать библиотеку в разрабатываемую систему.

1. Структура БД.
   1. Предметная область

В АИС используются следующие сущности:

* Пользователь
* Соискатель
* Работодатель
* Работник
* Задача
* Уровень образования
* Факультет

При регистрации создается сущность Пользователь. В зависимости от потребностей Пользователь может создать одну из двух сущностей. Если пользователь собирается искать тему для исследовательской работы или подработку, то он создает сущность Соискатель. Для этого необходимо заполнить поля в профиле. Если Пользователь является представителем организации, то он может создать сущность Работодатель, заполнив информацию об организации, в которой он работает. Пользователь, создавший Работодателя, автоматически становится Работником, привязанным к Работодателю.

К сущности Работодатель могут быть привязаны несколько Пользователей, которые будут отмечены как Работники. Добавлять к Работодателю новых Работников может только Пользователь, создавший Работодателя.

Работники могут создавать сущность Задача. Работник может курировать несколько Задач, и также за одной Задачей может быть закреплено несколько Работников.

Сущность Уровень образования обеспечивает ранжирование Задач по сложности и позволяет сопоставить уровень знаний Соискателя с требуемым.

Сущность Факультет описывает область знаний.

Отдельной сущности Администратор не предусмотрено. Подразумевается, что это Пользователь с максимальным уровнем доступа.

* 1. Логическая и физическая модель

На рисунке 1 изображена спроектированная логическая модель [6], построенная в diagrams.net.

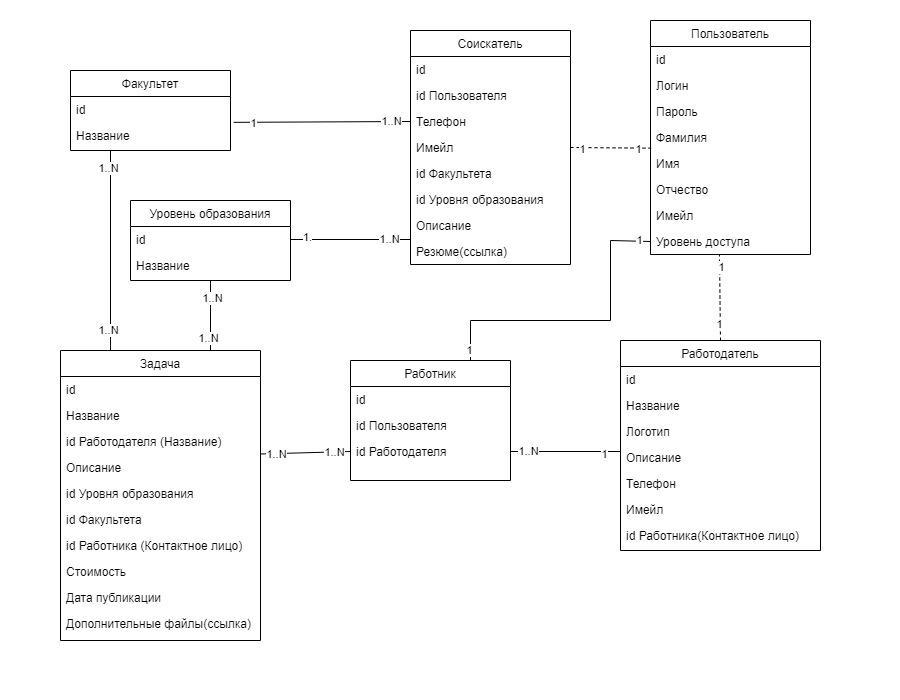


Рисунок 1 – Логическая модель БД

На логической модели отображено 7 основных таблиц – сущностей.

* 1. Описание сущностей БД

На основе логической модели построена физическая модель БД (см. Приложение), на которой кроме имен полей отображен их типы данных. Схема физической модели построена с помощью PostgreSQL.

На физической модели отображено 10 основных таблиц – сущностей и 3 дополнительные таблицы, которые поддерживают связь многие ко многим между основными сущностями.

Назначение сущности или же таблицы, образующие БД, описаны в таблице ниже.

Таблица 3 Описание сущностей

|  |  |  |
| --- | --- | --- |
| № | Имя сущности | Назначение сущности |
| 1 | user (Пользователь) | Сущность user используется для хранения свойств Пользователя |
| 2 | seeker (Соискатель) | Сущность seeker используется для хранения свойств Соискателя |
| 3 | employer (Работодатель) | Сущность employer используется для хранения свойств Работодателя |
| 4 | worker (Работник) | Сущность worker используется для хранения свойств Работника |
| 5 | worker\_has\_task | Промежуточная таблица, хранящая взаимоотношения Работников с Задачами |
| 6 | task (Задача) | Сущность task используется для хранения свойств Задачи |
| 7 | education (Уровень образования) | Сущность education используется для хранения возможных Уровней образования (бакалавр, магистр и т.д.) Соискателя и рекомендуемого уровня образования для выполнения Задачи |
| 8 | task\_has\_education | Промежуточная таблица, хранящая взаимоотношения задачи и возможных Уровней образования для её выполнения |
| 9 | faculty (Факультет) | Сущность faculty используется для хранения возможных Факультетов Задачи и Соискателя |
| 1 | task\_has\_faculty | Промежуточная таблица, хранящая взаимоотношения Задачи и возможных Факультетов для ее выполнения |

1. Программный продукт
   1. Авторизация

Страница «Авторизация» отображена ниже (см. рис.1).

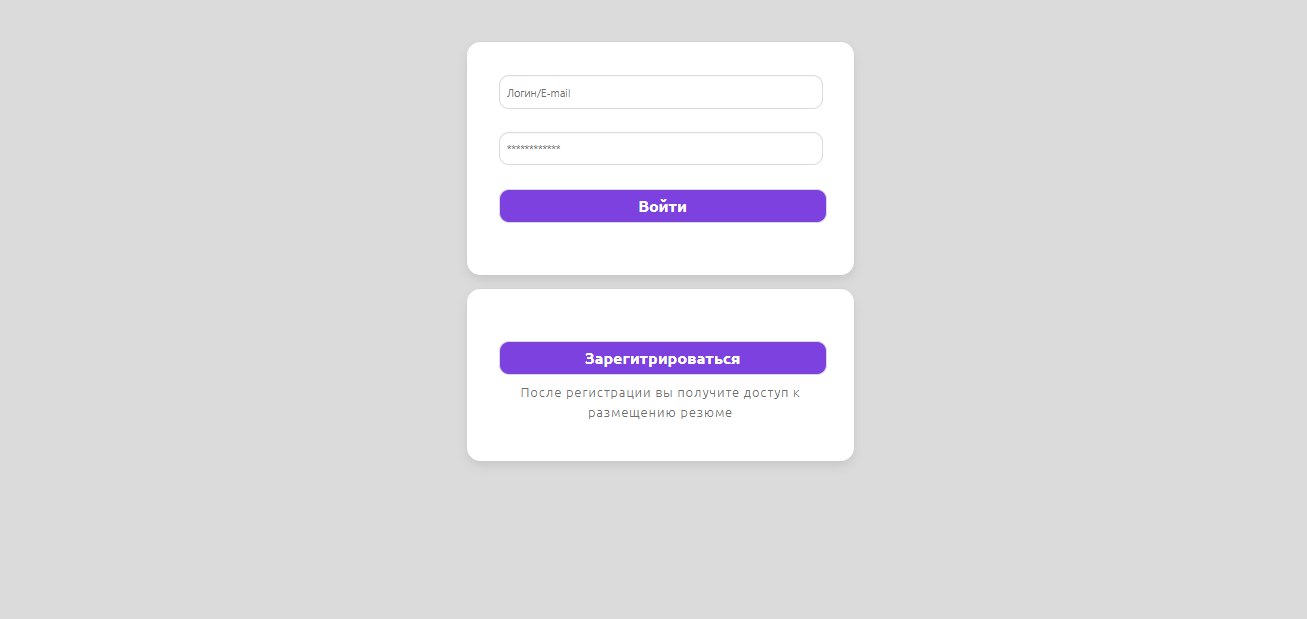


Рисунок 2 Страница Авторизации системы Upsilon Connect

Страница «Регистрации» Соискателя и Работодателя продемонстрирована на рисунке 2 и 3, соответственно. Различие состоит в том, что вместе с организацией Работодателя, регистрируется сотрудник их организации.

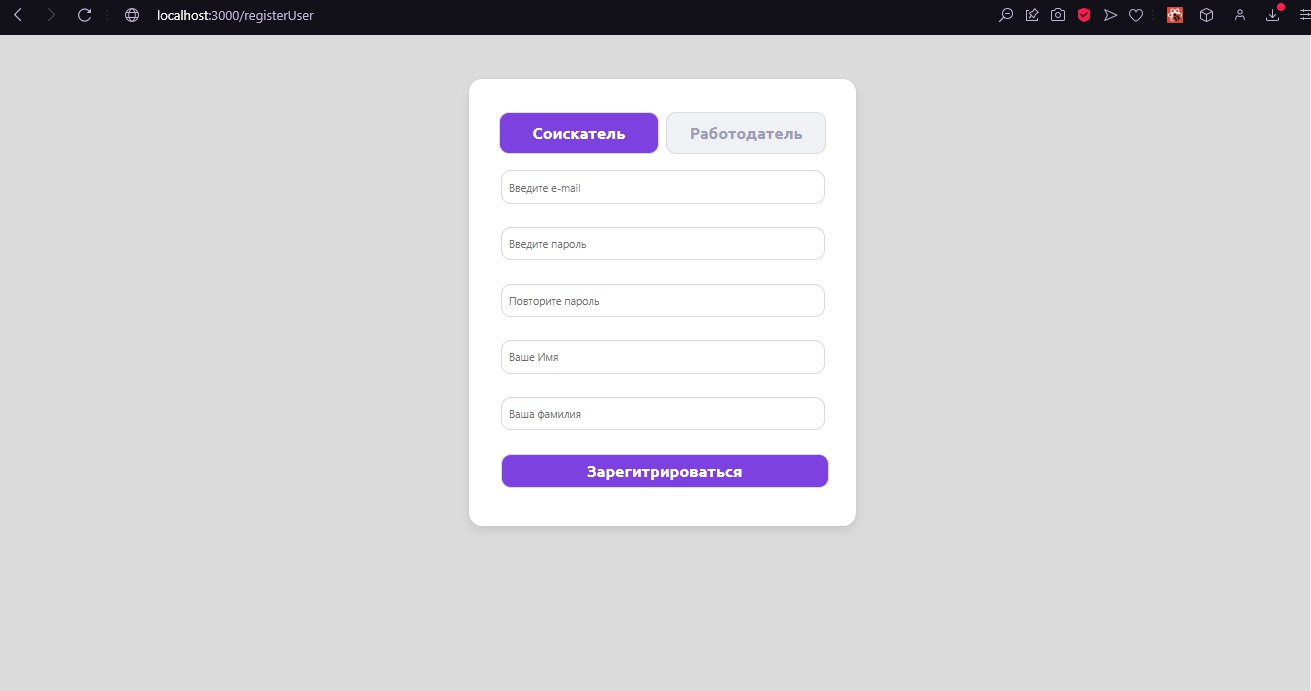
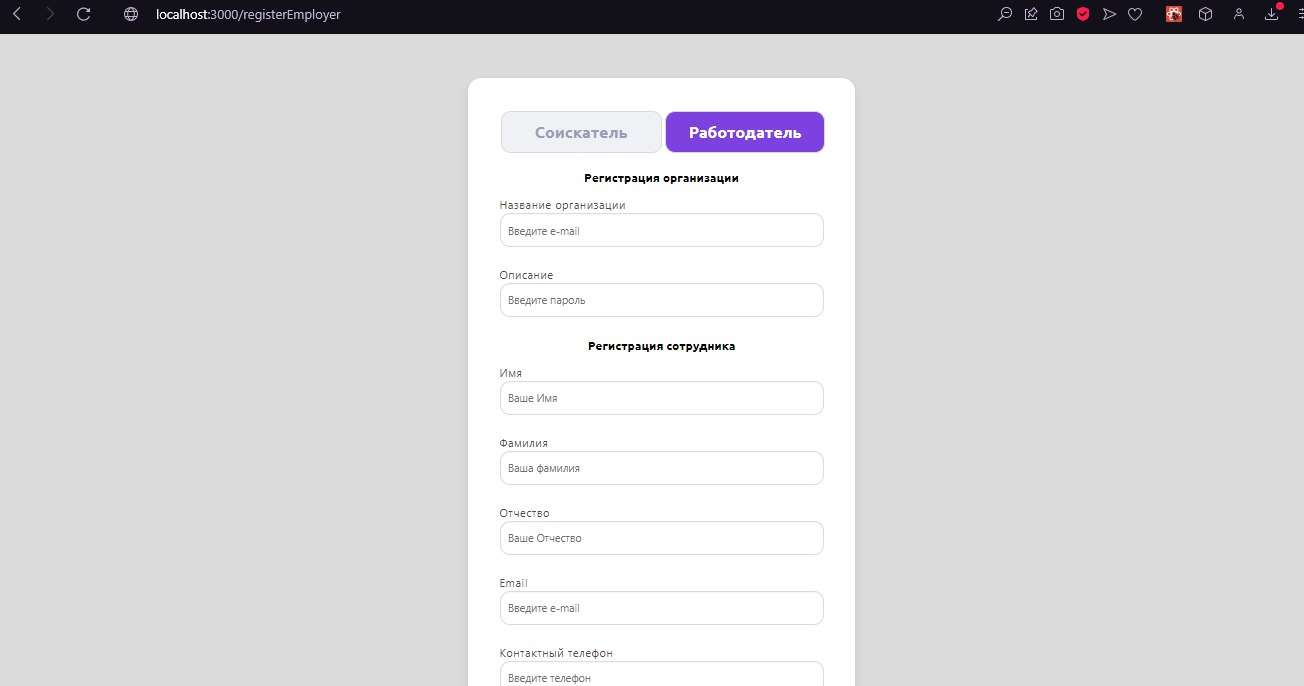


Рисунок 3 Форма регистрации Соискателя



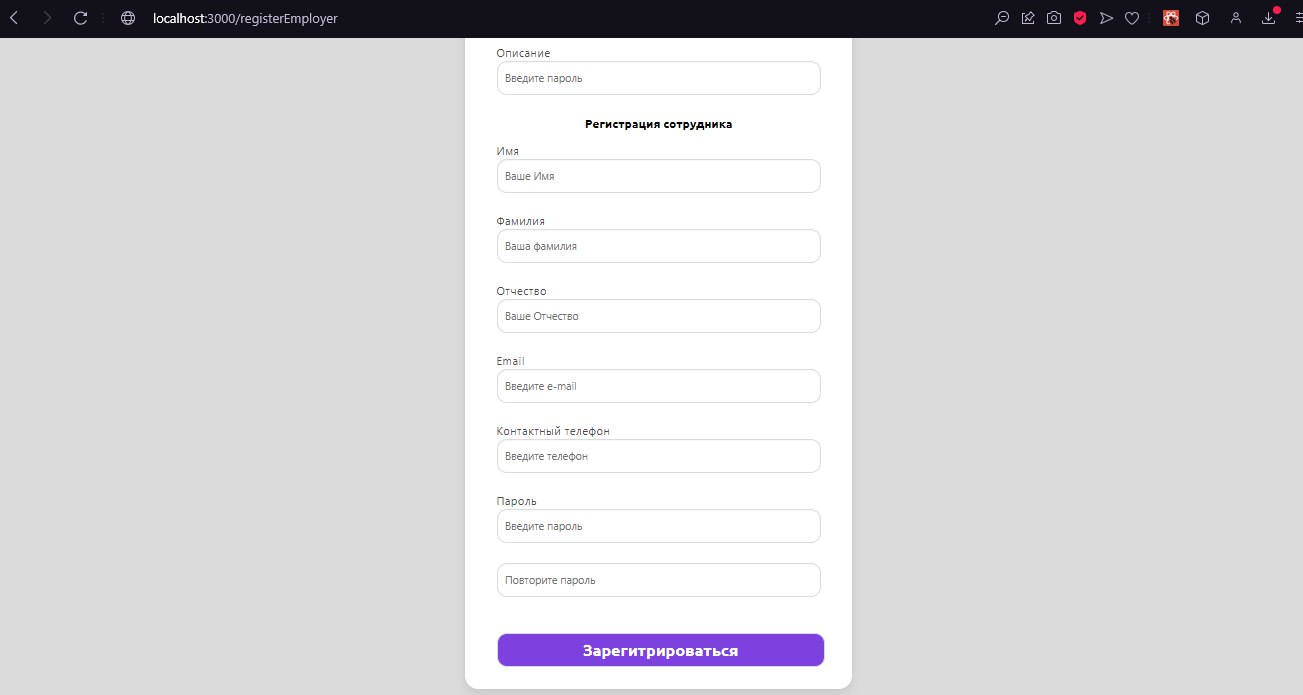


Рисунок 4 Форма регистрации Работодателя

* 1. Функционал ИС

На странице Задачи всем пользователям доступны для просмотра опубликованные карточки-задачи (см. рис.4). Код реализации представлен в Приложении 2.

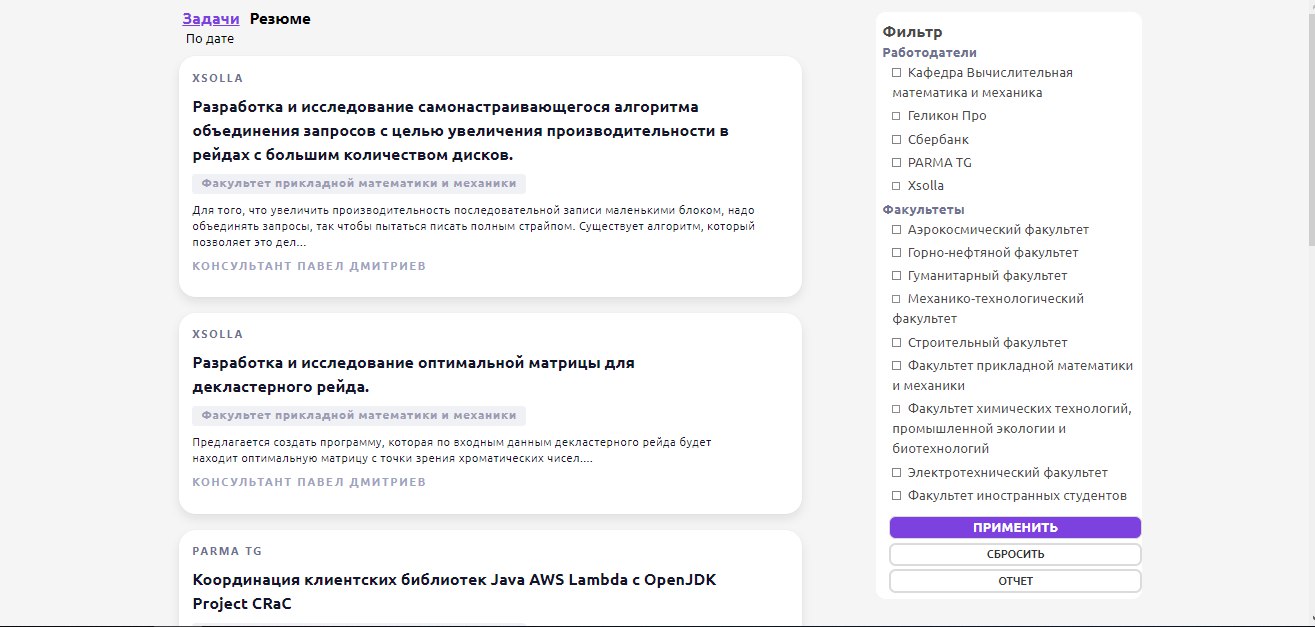


Рисунок 5 Страница Задачи

Пользователь имеет возможность отфильтровать задачи исходя из его предпочтений и специальности по категориям Факультеты и Работодатели, пример результатов предоставлен на рисунках 5 и 6. Также есть возможность отсортировать данные по дате публикации.

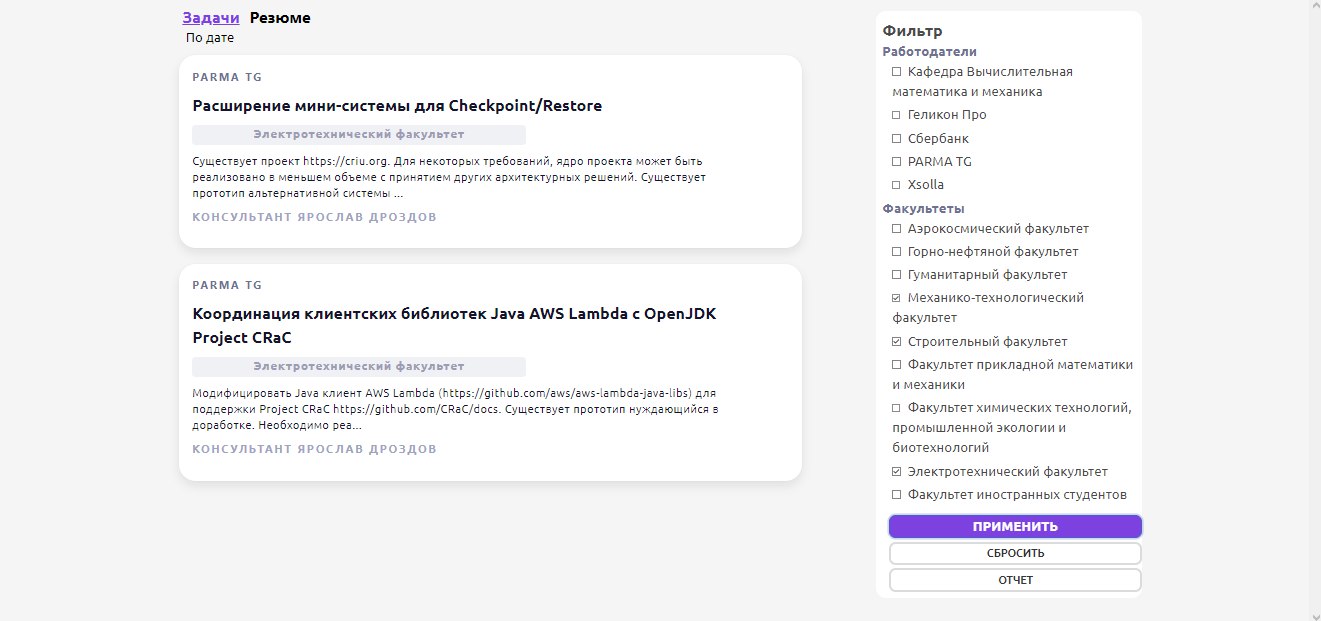


Рисунок 6 Вид страницы с использованием фильтрации по категории Факультет

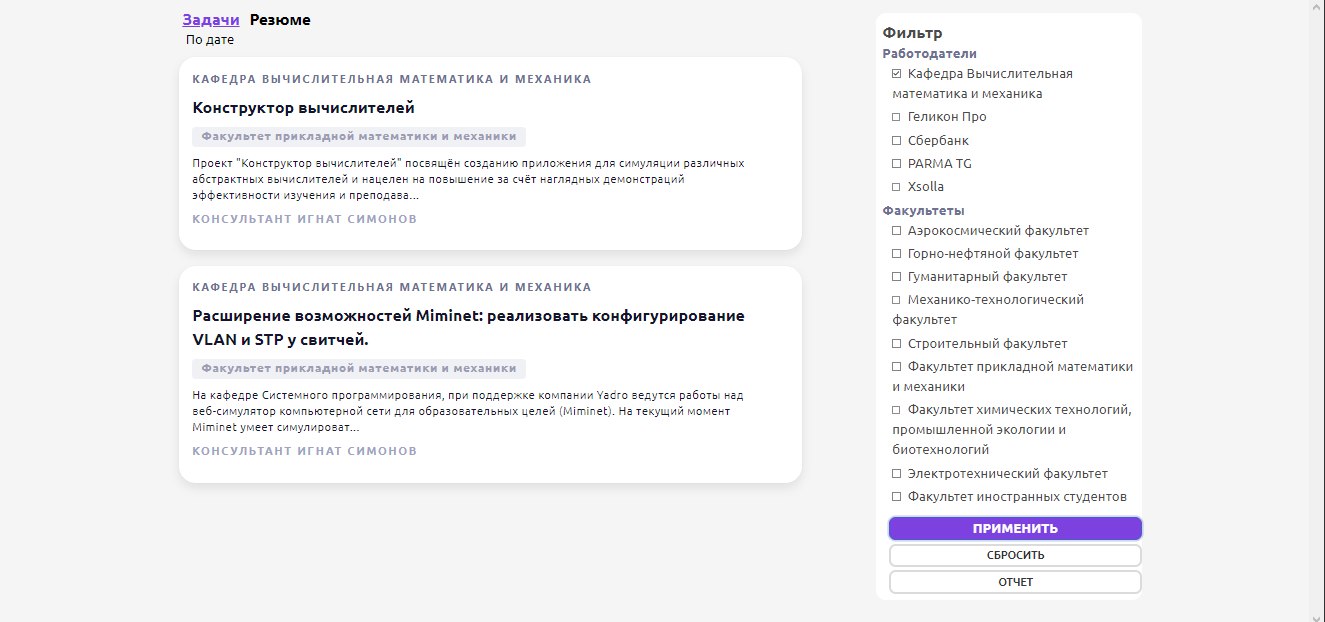


Рисунок 7 Пример страницы с использованием фильтрации по категории Работодатель

* 1. Отчет

Используя сортировку и фильтрацию, описанную в пункте 2.2.2, пользователь может выгрузить данные в отчете в формате .pdf. Результат отчета приведен на рисунке 7 и 8.

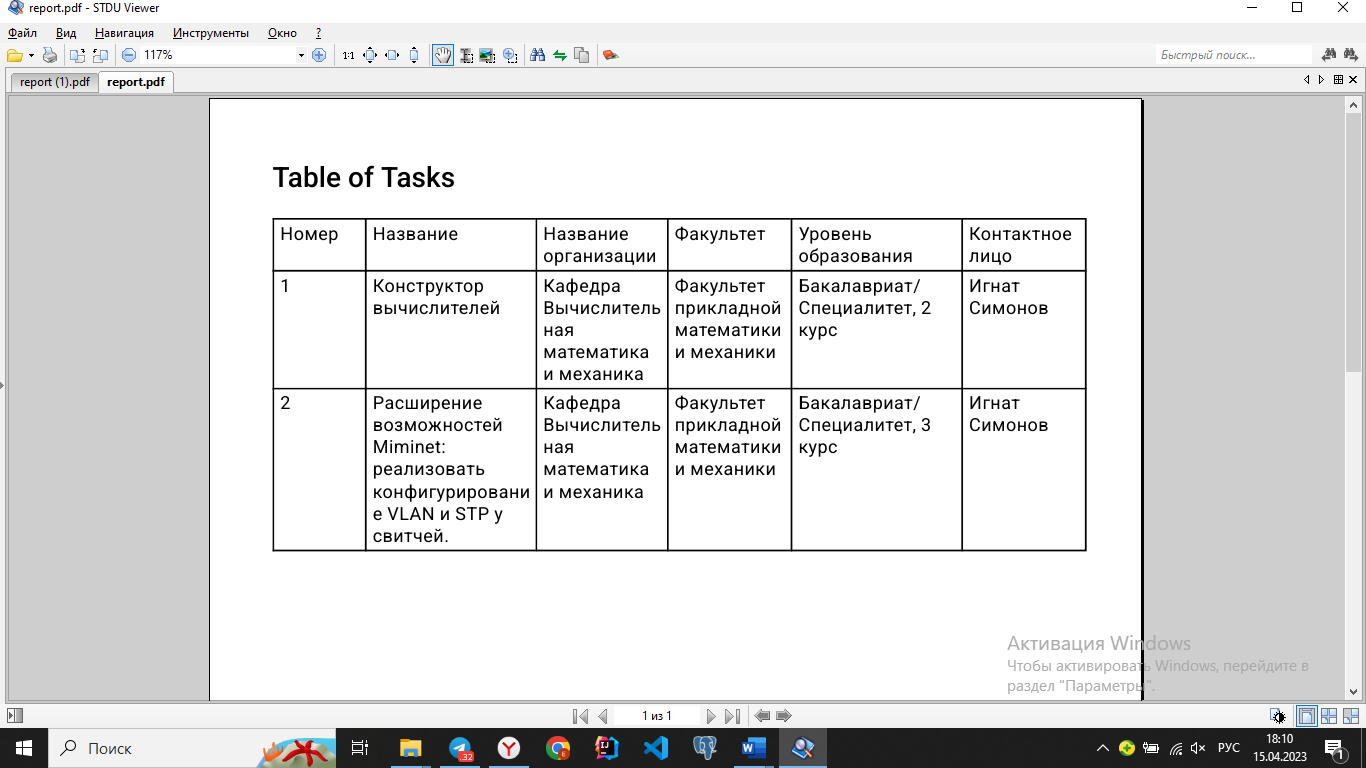


Рисунок 8 Результат отчета с фильтрацией по названию Организации- Кафедра ВММ

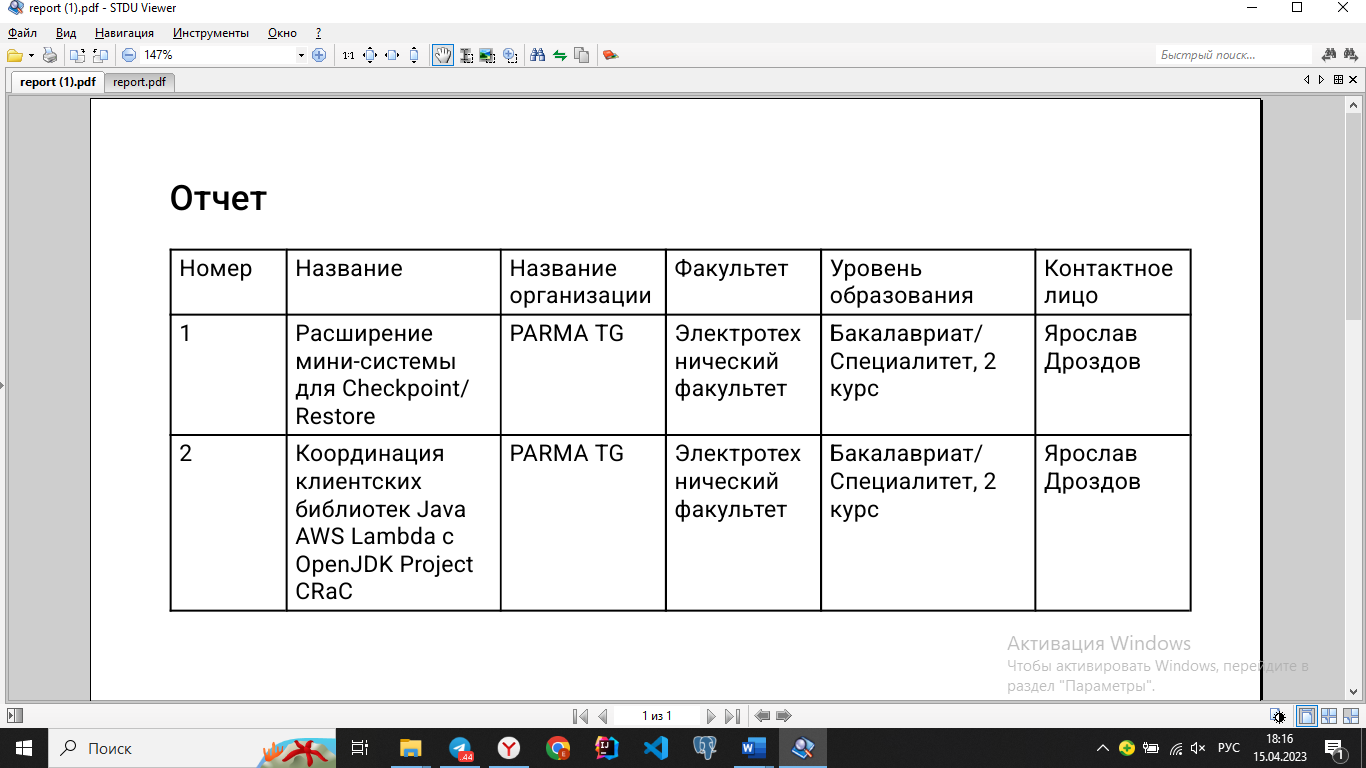


Рисунок 9 Результат отчета с применением фильтра по факультету- Электротехнический факультет

* 1. Реакция ИС на ошибочный ввод данных

В системе «Upsilon Connect» предусмотрены сценарии, когда пользователь запрашивает страницу, отсутствующую на сайте, то есть по запрошенному URL не соответствует ни одна страница сайта (рис. 9).

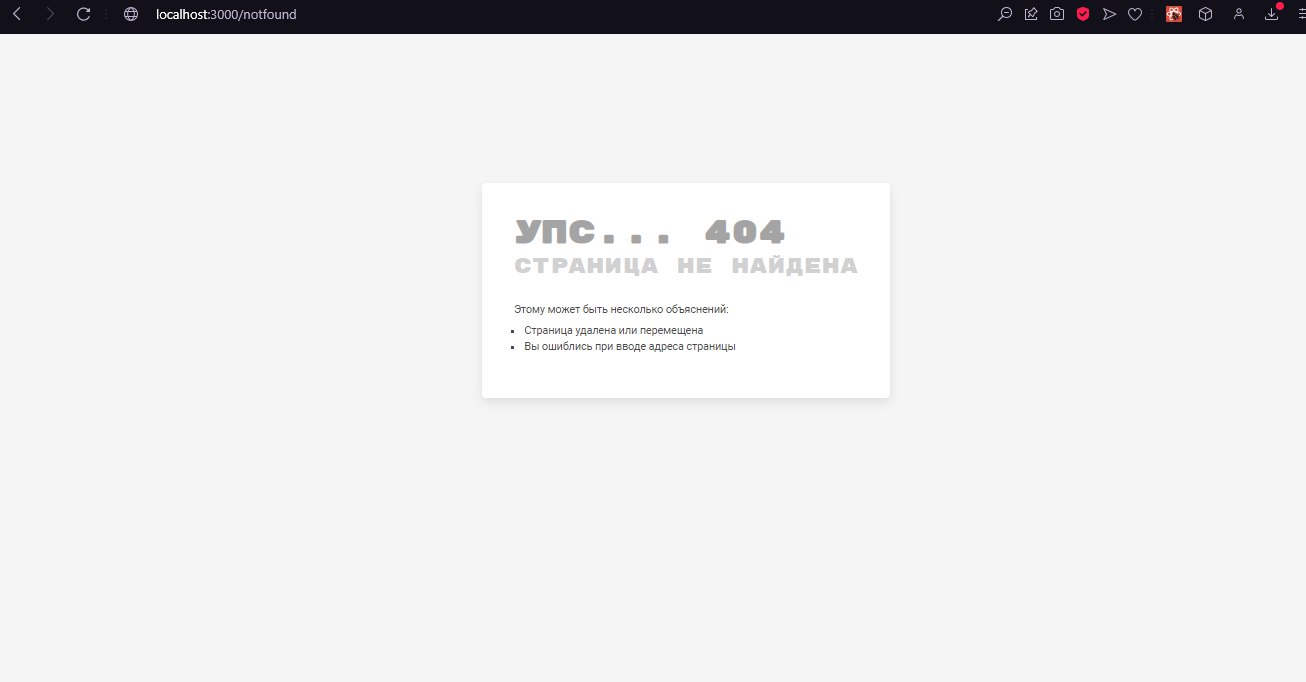


Рисунок 10 Страница с сообщением об ошибке 404

На странице Задачи при использовании фильтров возможен сценарий, когда не найдется результата, соответствующего пожеланием пользователя. В описанном случае пользователь увидит сообщение, уведомляющее пользователя о возникшей ошибке (рис.10).

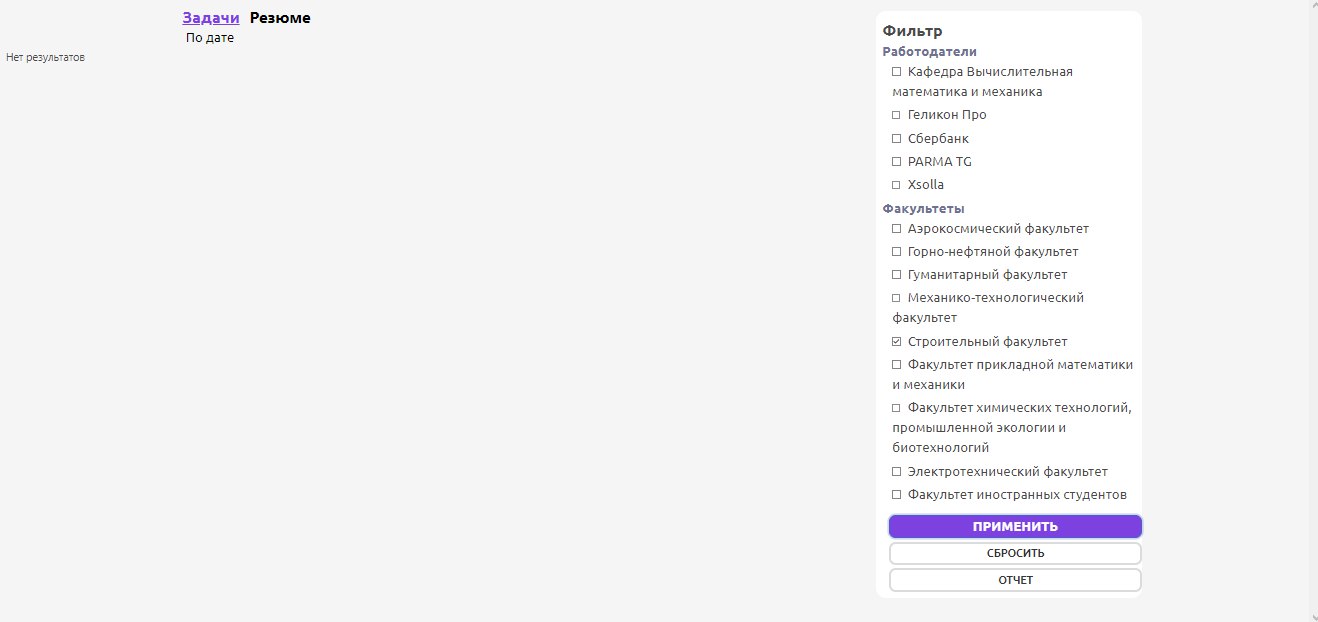


Рисунок 11 Ошибка "Нет результатов"

1. Выводы

В рамках работы была проанализирована предметная область и написано техническое задание для разработки. Было спроектировано хранилище данных, составлены логическая и физическая модели БД. На основе спроектированного интерфейса, созданной БД и разработанной структуры классов, написан программный код реализации.

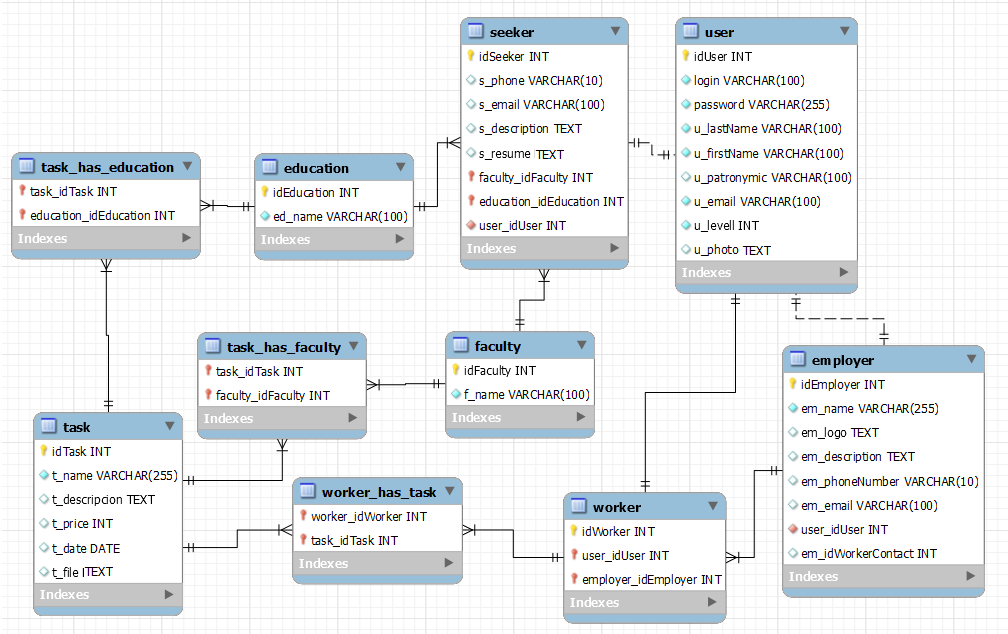
Система «Upsilon Contact», разработанная в рамках курсовой работы, позволяет пользователям найти научные и прикладные задачи, для дальнейшего сотрудничества, также генерировать необходимые отчеты.

Реализовав приложение с использованием программных компонентов, связанных с использованием базы данных, и информации, полученной из неё, можно отметить, что цель и задачи, определенные в начале исследования, достигнуты.

Список литературы

1. Какую СУБД выбрать и почему? (Статья 1): [Электронный ресурс] URL:https://habr.com/ru/articles/579248/ (Дата обращения: 14.04.2023).
2. Миковски М. С.: Разработка одностраничных веб-приложений: Пауэлл Джош К.— М.: ДМК-Пресс, 2018, —512с.
3. Тиленс Т.Н. React в действии: -Издательский Дом ПИТЕР, 2019. —368 с.
4. [в закладки] PDF- и ePUB-версия руководства по React**:** [Электронный ресурс] URL: https://habr.com/ru/companies/ruvds/articles/471040/(Дата обращения: 14.04.2023).
5. jsPDF Documentation: [Электронный ресурс] URL: https://gaam-akhar.ir/assets/jspdf/docs/jsPDF.html/(Дата обращения: 14.04.2023).
6. Б. Новиков Основы технологий баз данных: учебное пособие: Е. Горшкова, Н. Графеева. — М.: ДМК-Пресс, 2020, —582с.

Приложение 1



Приложение 2

Страница Задачи. Директория Task.

Файл index.jsx

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import Tasks from './task';

import FilterForm from './filter';

import generatePDF from '../../app/components/ReportGenerator/reportGenerator';

function FilterPage() {

const [filteredTasks, setFilteredTasks] = useState([]);

const [tasks, setTasks] = useState([]);

useEffect(() => {

axios.post('http://localhost:8080/api/task')

.then(response => {

setTasks(response.data);

setFilteredTasks(response.data)

})

.catch(error => {

//console.log(error);

});

}, []);

const handleGeneratePDF = () => {

console.log(filteredTasks);

generatePDF(filteredTasks); // Вызываем функцию generatePDF и передаем ей пропсы из родительского компонента

};

return (

<div className='tasks'>

<FilterForm setFilteredTasks={setFilteredTasks} onClick={handleGeneratePDF} />

{filteredTasks.length > 0 ? (

<Tasks tasks={filteredTasks} />

) : (

<p>Нет результатов</p>

)}

</div>

);

}

export default FilterPage;

Файл task.jsx

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import TaskCard from '../../app/components/Task';

function Tasks(props) {

const [tasks, setTasks] = useState([]);

return (

<div className='tasks'>

{Array.from(props.tasks).map(task => (

<TaskCard key={task.id} task={task}/>

))}

</div>

);

}

export default Tasks;

Файл filter.jsx

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import './filter.scss';

import { Button } from 'react-bulma-components';

import generatePDF from '../../app/components/ReportGenerator/reportGenerator';

function FilterForm(props) {

const [employers, setEmployers] = useState([]);

const [faculties, setFaculties] = useState([]);

const [filters, setFilters] = useState([]);

useEffect(() => {

// Получение данных из базы данных через axios

axios.get('http://localhost:8080/api/employer').then(response => setEmployers(response.data));

axios.get('http://localhost:8080/api/faculty').then(response => setFaculties(response.data));

}, []);

const handleFilterChange = event => {

const { name, checked } = event.target;

if (checked) {

setFilters([...filters, name]);

} else {

setFilters(filters.filter(filter => filter !== name));

}

};

const handleSubmit = event => {

event.preventDefault();

const employerIds = filters.filter(filter => filter.startsWith('employer\_')).map(filter => filter.split('\_')[1]);

const facultyIds = filters.filter(filter => filter.startsWith('faculty\_')).map(filter => filter.split('\_')[1]);

const data = {

employer\_ids: employerIds,

faculty\_ids: facultyIds

};

axios.post('http://localhost:8080/api/task/filter', data).then(response => {

props.setFilteredTasks(response.data);

});

};

const handleGeneratePDF = () => {

console.log(props);

generatePDF(props.filteredTasks); // Вызываем функцию generatePDF и передаем ей пропсы из родительского компонента

};

return (

<div>

<nav>

<a className="flex1" href="#">Задачи</a>

<a className="flex" href="#">Резюме</a>

</nav>

<div>

<a className="hea">По дате</a>

</div>

<form className='filtr' onSubmit={handleSubmit}>

<h1 className='filtre'>Фильтр</h1>

<div>

<h3 className='filter-name'>Работодатели</h3>

{employers.map(employer => (

<label key={employer.em\_id}>

<input className='input-filter' type="checkbox" name={`employer\_${employer.em\_id}`} onChange={handleFilterChange} />

{employer.em\_name}

</label>

))}

</div>

<div>

<h3 className='filter-name'>Факультеты</h3>

{faculties.map(faculty => (

<label key={faculty.f\_id}>

<input className='input-filter' type="checkbox" name={`faculty\_${faculty.f\_id}`} onChange={handleFilterChange} />

{faculty.f\_name}

</label>

))}

</div>

<Button type = "submit" className='Applys'>Применить</Button>

<Button className='Clears'>Сбросить</Button>

<Button className='Othet' onClick={props.onClick} >Отчет</Button>

</form>

</div>

);

}

export default FilterForm;

Файл filter.scss

.flex1{

right: 25px;

cursor: pointer;

font-size: 24px;

font-weight: bold;

font-family: 'Ubuntu', sans-serif;

bottom: 20px;

padding: 15px;

bottom: 20px;

color: #7C41DE;

margin-left: 250px;

text-decoration: underline;

}

.flex{

right: 25px;

cursor: pointer;

font-size: 24px;

bottom: 20px;

padding-right: 15px;

bottom: 20px;

color: black;

font-weight: bold;

}

.hea{

margin-left: 250px;

margin-top: 25px;

padding: 20px;

font-family: 'Ubuntu', sans-serif;

font-size: 20px;

color: black;

}

.filtr{

position: fixed;

right: 250px;

top: 20px;

padding: 10px;

width: 400px;

border-radius: 15px;

background-color: white;

}

.filtre{

font-size: 24px;

font-weight: bold;

font-family: 'Ubuntu', sans-serif;

}

.filter-name

{

font-size: 20px;

font-weight: bold;

font-family: 'Ubuntu', sans-serif;

color: #6E7191;

}

label{

display:block;

font-size:20px;

font-family: 'Ubuntu', sans-serif;

font-weight:500;

padding: 0 0 5px 15px;

}

.tasks{

border-radius: 3%;

margin: 10px;

}

.Applys{

width:100%;

height: 35px;

margin-left: 10px;

margin-top: 10px;

background: #7C41DE;

color: white;

font-size:20px;

font-family: 'Ubuntu', sans-serif;

font-weight:bold;

border-radius: 10px;

text-transform: uppercase;

}

.Applys:hover

{

color:#fff;

}

.Applys:active

{

color:#fff;

}

.Applys:focus

{

color:#fff;

}

.Clears{

width:100%;

height: 35px;

margin-left: 10px;

margin-top: 5px;

background: #fff;

border-width: 4px;

border-radius: 10px;

font-family: 'Ubuntu', sans-serif;

font-weight:bold;

text-transform: uppercase;

}

.Othet{

width:100%;

height: 35px;

margin-left: 10px;

margin-top: 5px;

background: #fff;

border-width: 4px;

border-radius: 10px;

font-family: 'Ubuntu', sans-serif;

font-weight:bold;

text-transform: uppercase;

}

input[type=checkbox], input[type=radio] {

margin-right: 10px!important;

}

Компонент генерации отчетов. Директория app/components/ ReportGenerator.

Файл reportGenerator.jsx

import pdfMake from "pdfmake/build/pdfmake";

import pdfFonts from "pdfmake/build/vfs\_fonts";

pdfMake.vfs = pdfFonts.pdfMake.vfs;

function generatePDF(props) {

const tasks = props;

const tableBody = tasks.map((task, index) => [

index + 1,

task.name,

task.employer,

task.faculty,

task.education,

`${task.workername} ${task.workersurname}`,

]);

const documentDefinition = {

content: [

{ text: "Отчет", style: "header" },

{

style: "tableExample",

table: {

widths: [50,100, 75, 70, 100, 70],

headerRows: 1,

body: [[

"Номер",

"Название",

"Название организации",

"Факультет",

"Уровень образования",

"Контактное лицо",

], ...tableBody],

},

},

],

styles: {

header: {

fontSize: 18,

bold: true,

margin: [0, 0, 0, 10],

},

tableExample: {

margin: [0, 5, 0, 15],

},

},

};

pdfMake.createPdf(documentDefinition).download("report.pdf");

}

export default generatePDF;