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»

**ПРАКТИЧЕСКАЯ РАБОТА**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Программная реализация КИС»**

Выполнили:

студенты гр. ИСТ-19-1Б

Ким Елизавета Алексеевна

Решетникова Елизавета Андреевна

Лопина Мария Сергеевна

(Ф.И.О.)

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

*(подпись)*

Принял:

ст. преподаватель, Банников Р.Ю.

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

*(должность, ФИО руководителя)*

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

*(оценка) (подпись)*

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

*(дата)*

**Пермь 2023**

Оглавление

[**1.** Постановка задачи 3](#_Toc132477681)

[2. Реализация 5](#_Toc132477682)

[3. Вывод 10](#_Toc132477683)

[4. Приложение 11](#_Toc132477684)

# 

## Постановка задачи

* 1. Описание назначения КИС и основного функционала согласно ТЗ.

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

* 1. 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.
  1. 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.
  1. Текущее состояние объекта автоматизации

На текущий момент взаимодействие работодателя и исполнителя по поиску и выбору задачи осуществляется вручную без фиксации контроля со стороны работодателя действий исполнителя. В связи с чем данные получаются некорректными.

* 1. Основной функционал ИС

Система «Upsilon Contact» предоставляет возможность пользователям, заинтересованных в решении научных и прикладных задач и подработке, найти её. А компаниям, преподавателям и другим заинтересованным лицам (Работодатели) найти Соискателей, готовых решить их задачи.

В функционал описываемой ИС входят передача и хранение информации о Задачах, выдача этой информации пользователю, а также выборка по запросу пользователя. Выборка по данным предоставляется пользователям в виде отчета по критериям, заданным в фильтре на странице Задачи. В информационной системе предусмотрена регистрация и авторизация пользователей.

* 1. Цель работы и задачи

Цель: Реализация информационной системы, удовлетворяющий функционалу, описанному в ТЗ.

Задачи:

Выбрать средства разработки, фреймворки, библиотеки и среду разработки.

Реализовать функционал согласно ТЗ:

* 1. Авторизация и регистрация;
  2. Страница Задачи с фильтрацией и кнопкой выгрузкой отчета;
  3. Реализация вывода отчета;
  4. Реакция системы на ошибочный ввод данных;

Проанализировать конечную работу.

## Реализация

* 1. Средства реализации программного кода

Программный код реализован под средством языкового программирования JavaScript и фреймворка React в среде Visual Studio Code. Для автоматизации формирования отчета была подключена библиотека jsPDF.

* 1. Работа основного функционала
     1. Авторизация

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

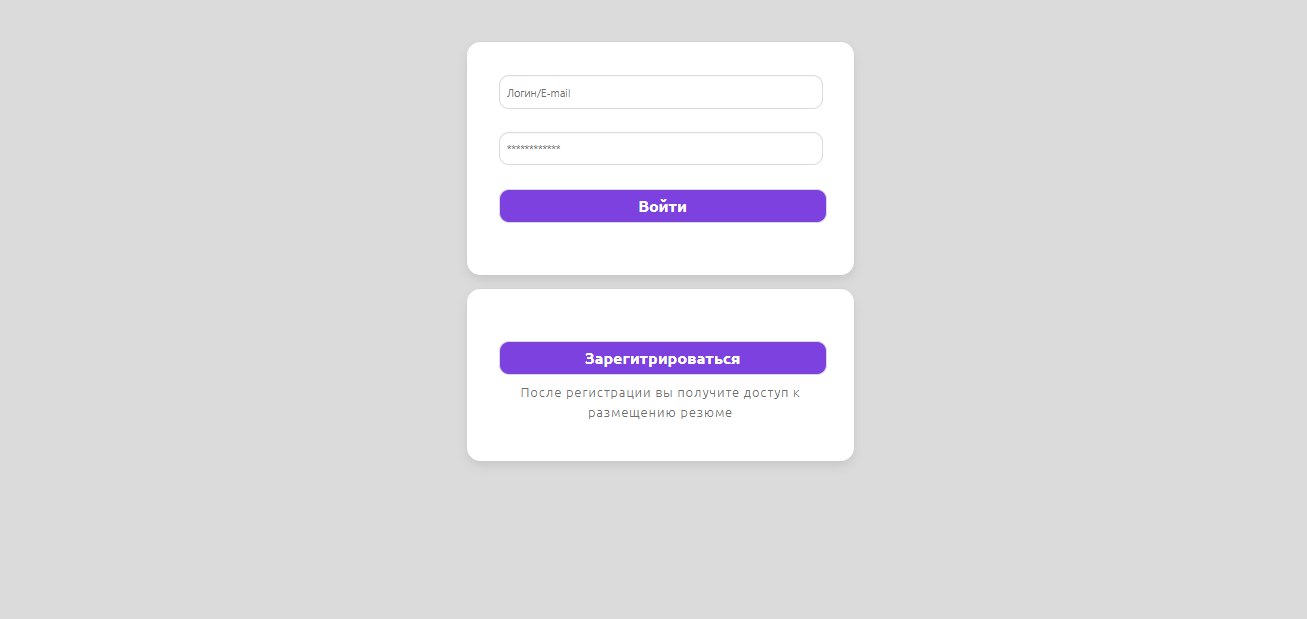


Рисунок 1 Страница Авторизации системы Upsilon Connect

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.

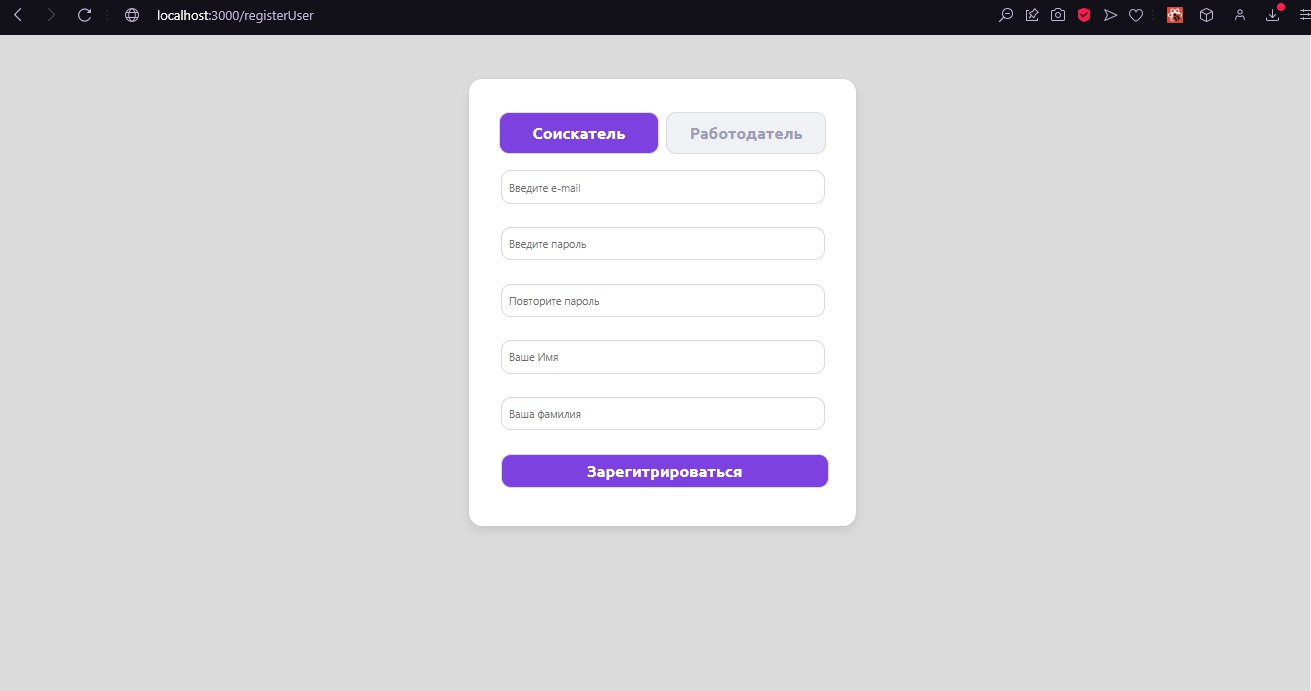
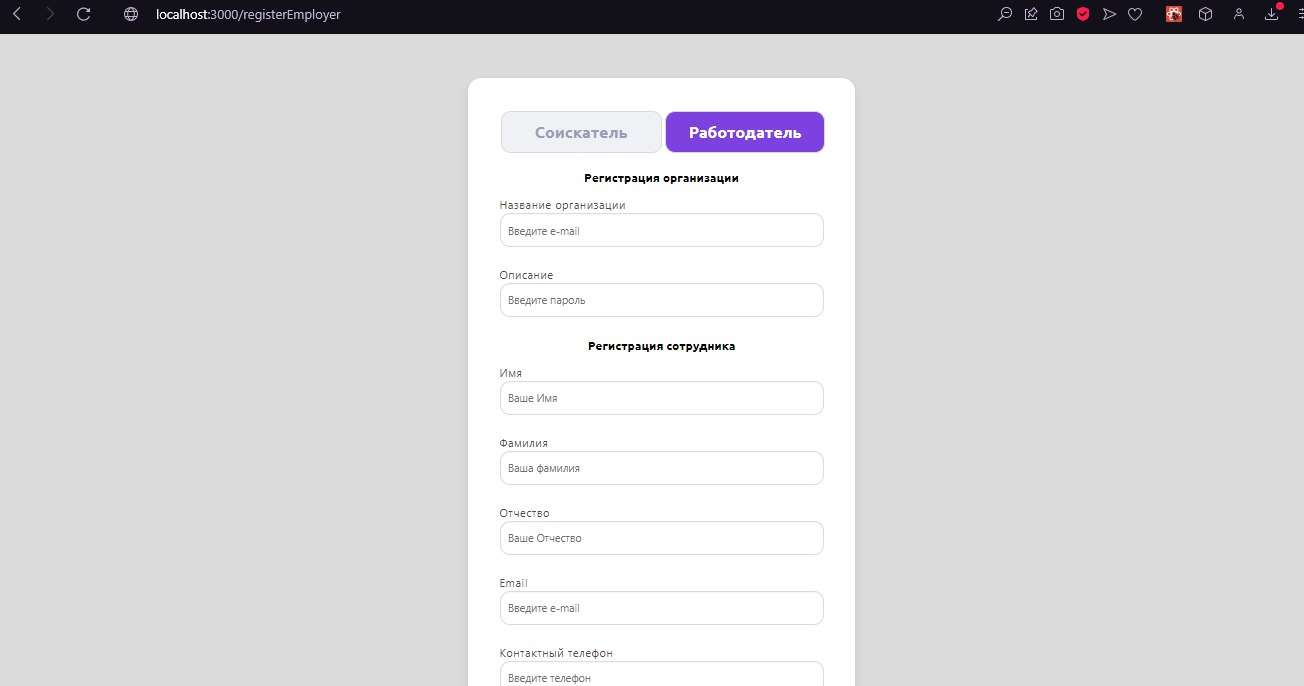


Рисунок 2 Форма регистрации Соискателя



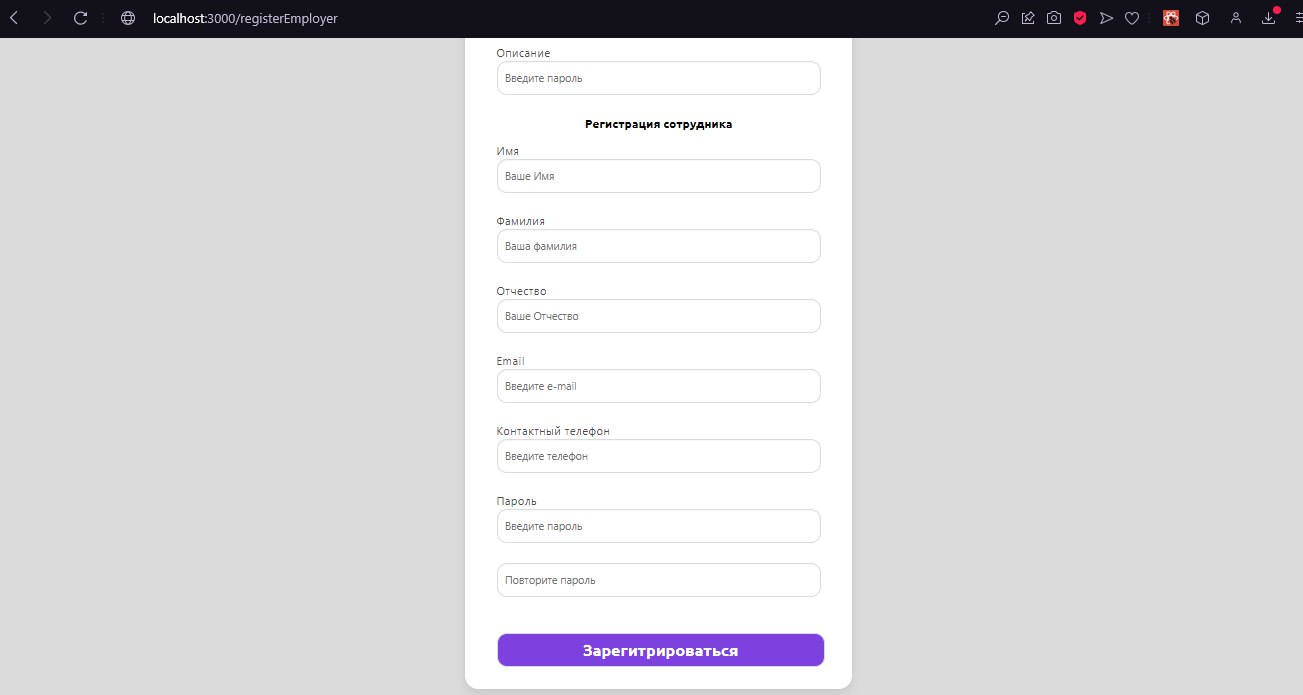


Рисунок 3 Форма регистрации Работодателя

* + 1. Функционал ИС

На странице Задачи всем пользователям доступны для просмотра опубликованные карточки-задачи (см. рис.4).

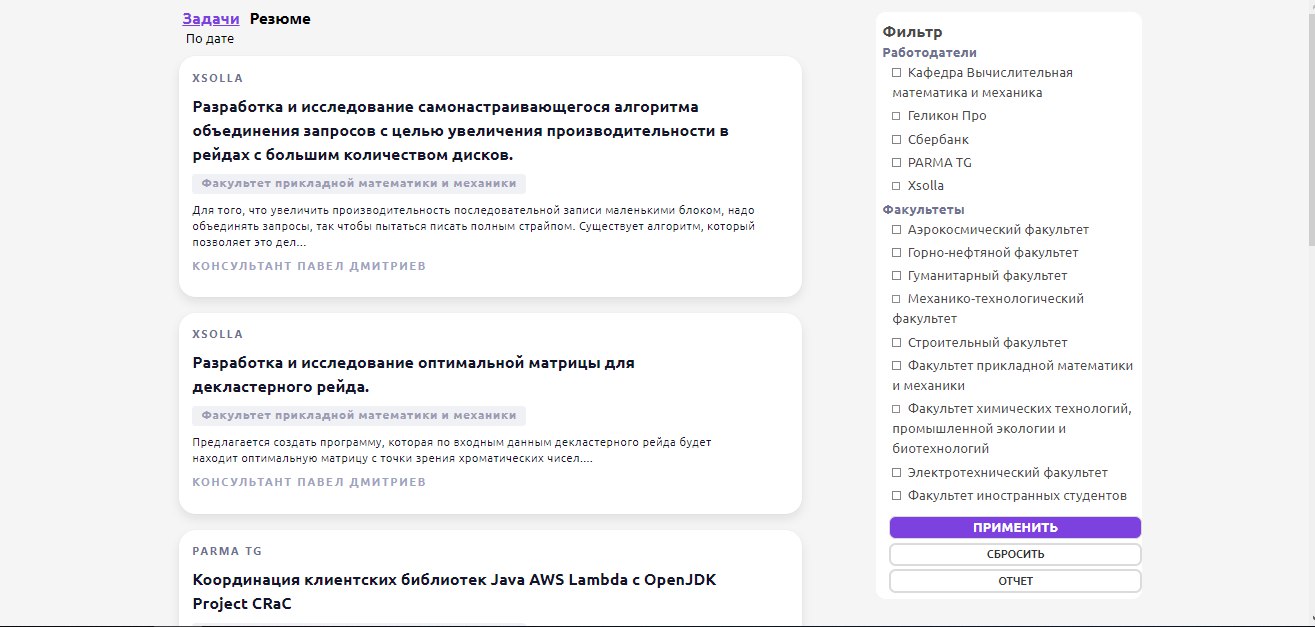


Рисунок 4 Страница Задачи

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.

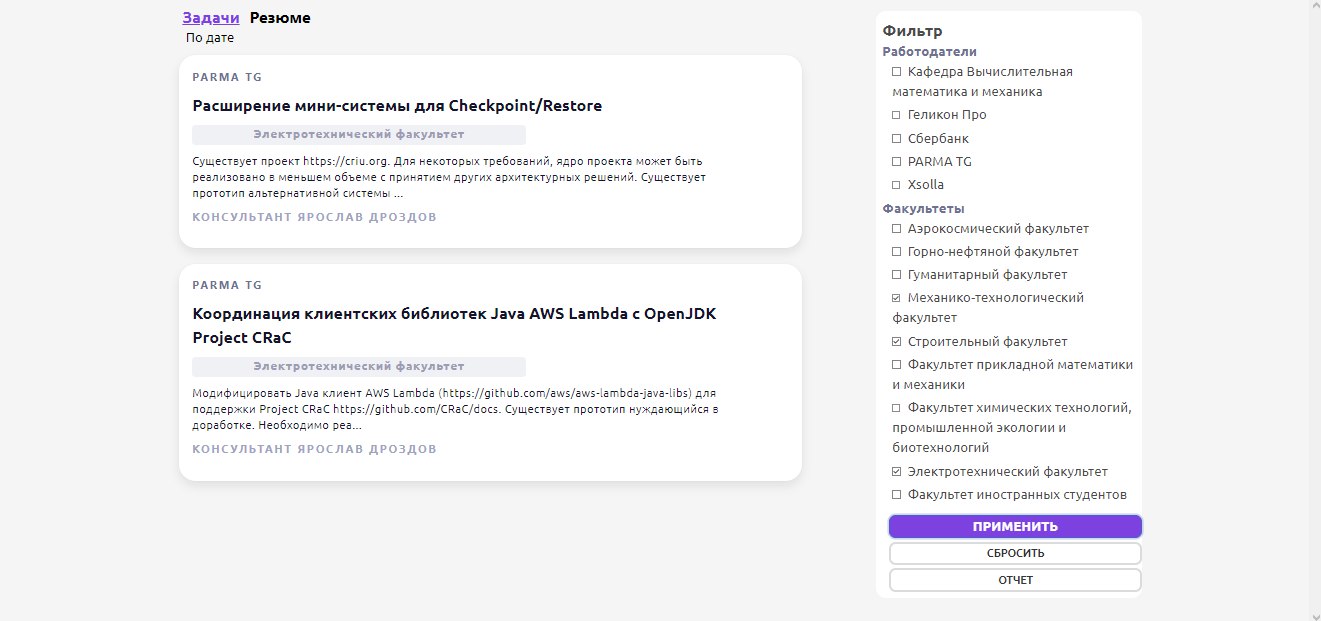


Рисунок 5 Вид страницы с использованием фильтрации по категории Факультет

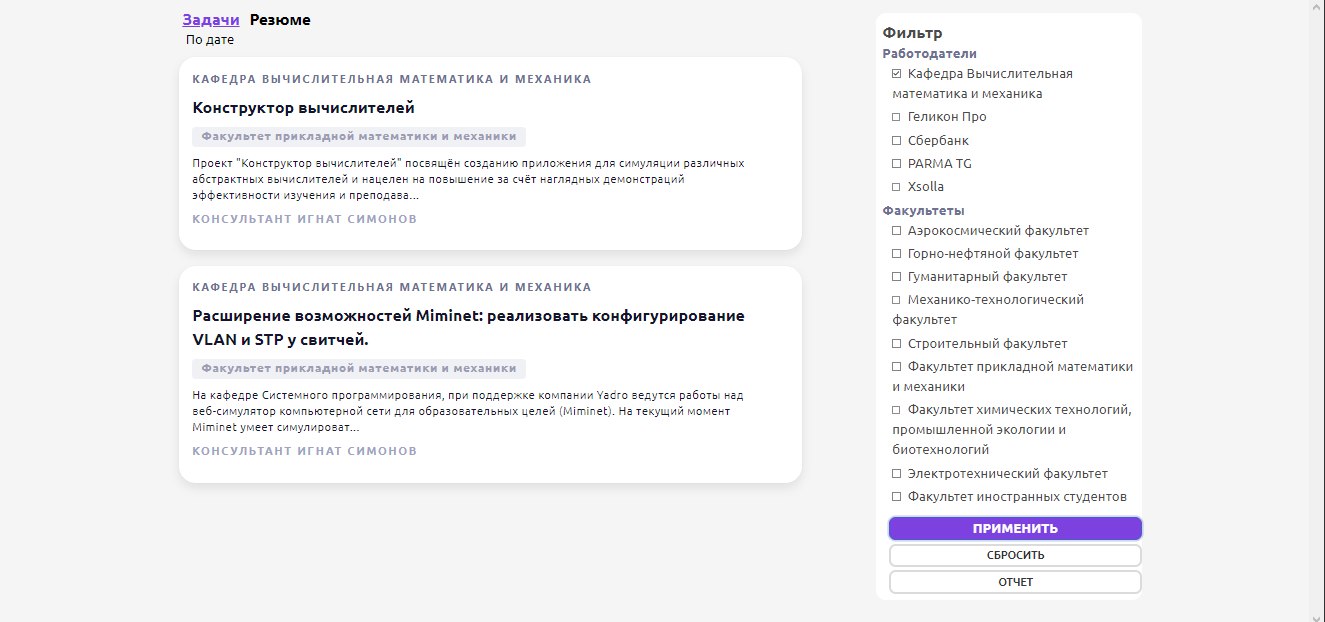


Рисунок 6 Пример страницы с использованием фильтрации по категории Работодатель

* + 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.

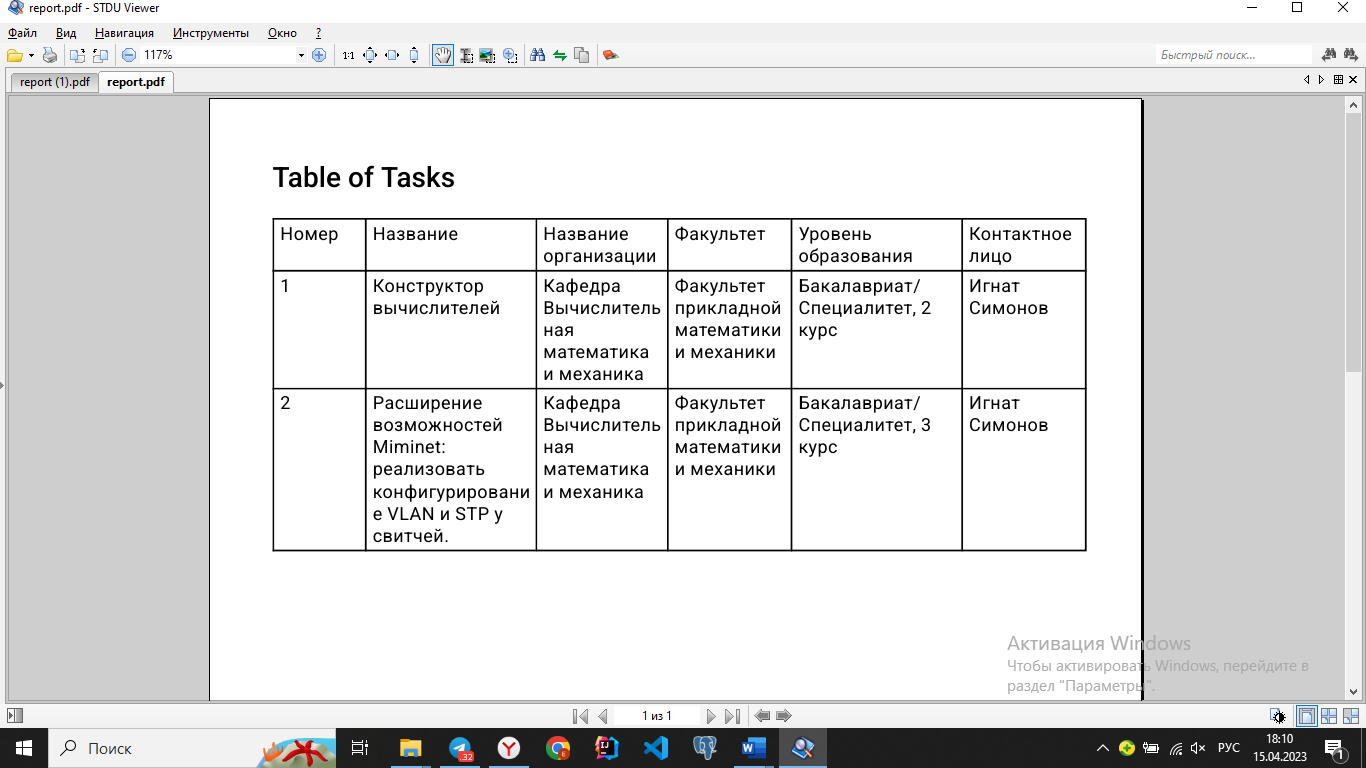


Рисунок 7 Результат отчета с фильтрацией по названию Организации- Кафедра ВММ

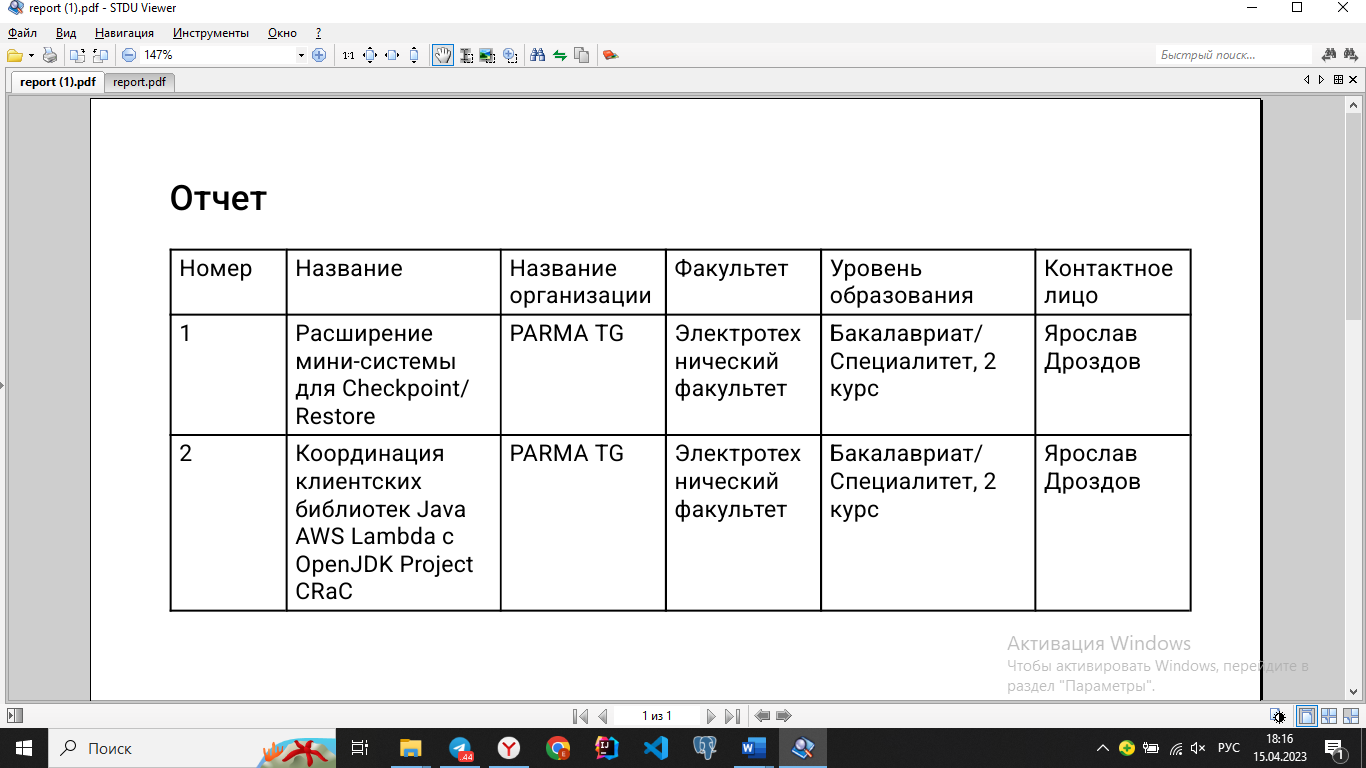


Рисунок 8 Результат отчета с применением фильтра по факультету- Электротехнический факультет

* + 1. Реакция ИС на ошибочный ввод данных

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

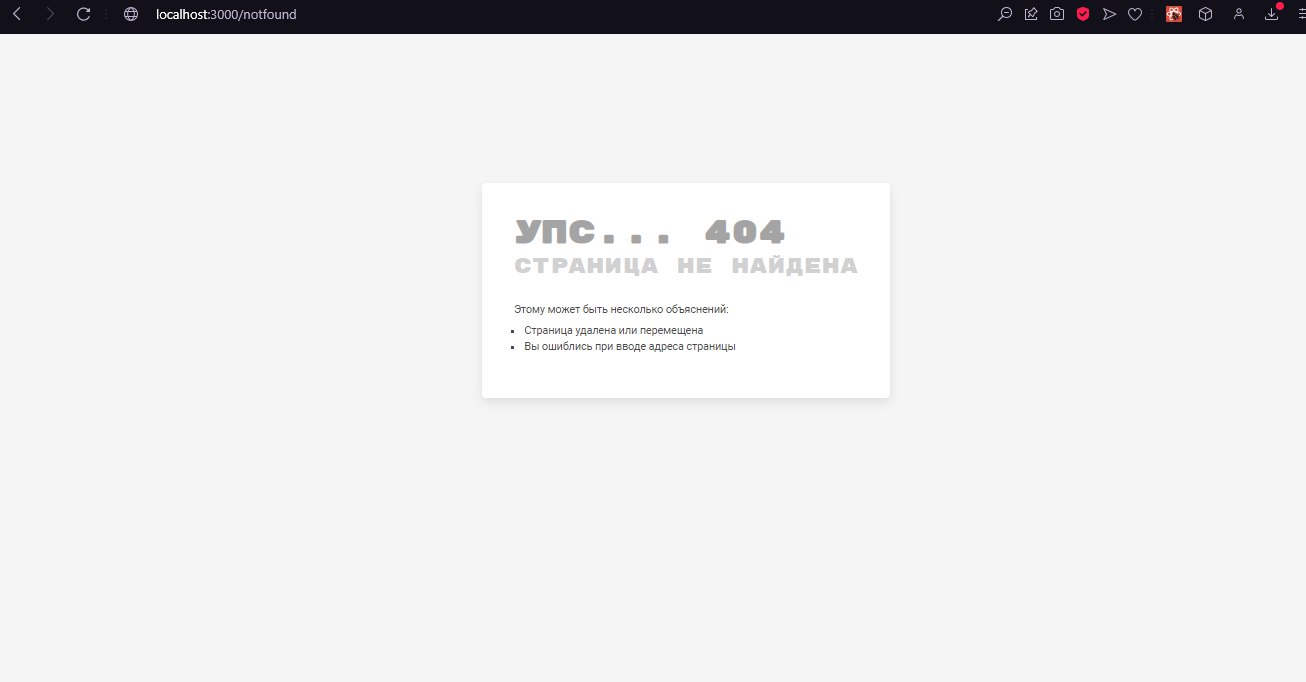


Рисунок 9 Страница с сообщением об ошибке 404

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).

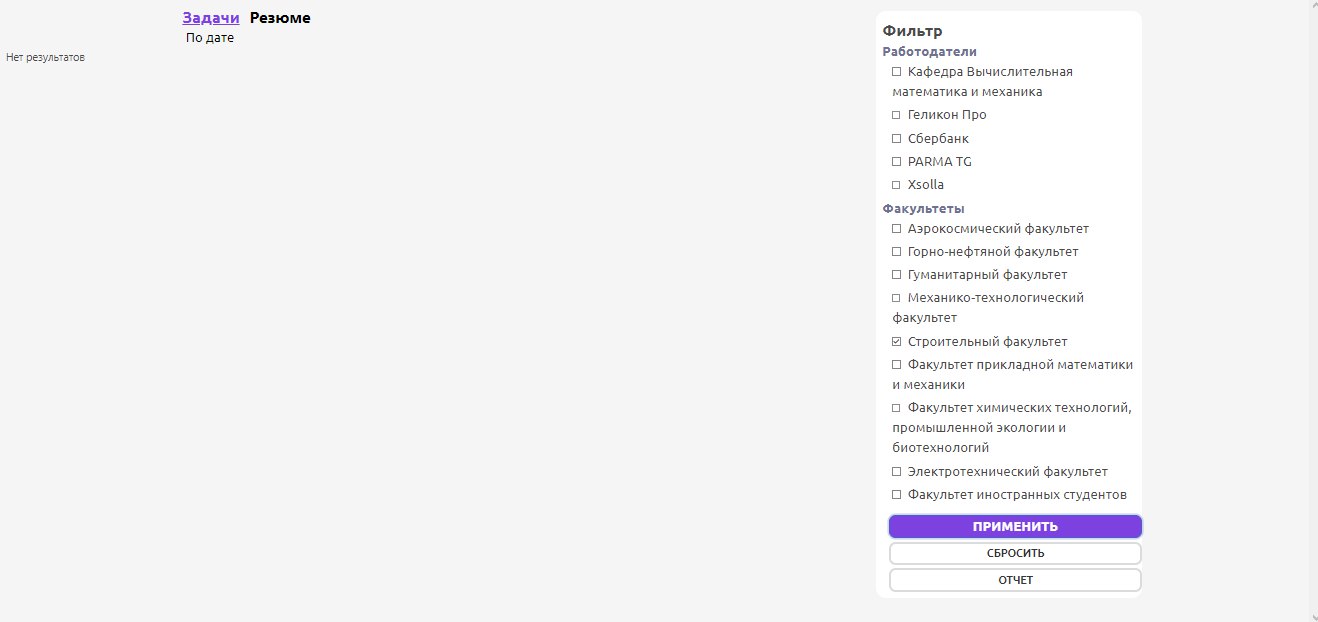


Рисунок 10 Ошибка "Нет результатов"

## Вывод

В рамках практических занятий была реализована информационная система, с описанным в ТЗ функционалом, с спроектированной базы данных, разработанным интерфейсом и структурой классов. Но не весь функционал, согласно ТЗ, удалось исполнить. Такой функционал как Авторизация, редактирование данных, содержащих в системе и функционал ИС под разными пользователями планируется в последующей разработке. В дальнейшем предстоит тестирование, разработанной системы.

## Приложение

Структура системы:

* upsilon
* public
* src
  + app
    - components
      * Header
        + Header.scss
        + index.jsx
      * ReportGenerator
        + reportGenerator.js
      * Rezume
        + Rezume.scss
        + index.jsx
      * Task
        + index.jsx
        + taskCard.scss
    - controller
      * rezumeController.js
      * taskController.js
      * userController.js
    - router
      * userRoutes.js
    - store
      * store.js
    - App.js
    - App.scss
    - Constants.js
    - Db.js
    - Router.jsx
  + pages
    - Login
      * LoginPage.scss
      * index.jsx
    - NotFound
      * NotFound.scss
      * index.jsx
    - RegisterEmployer
      * RegisterPageEmployer.scss
      * index.jsx
    - RegisterUser
      * RegisterPageUser.scss
      * index.jsx
    - Rezume
      * index.jsx
    - Task
      * filter.jsx
      * filter.scss
      * index.jsx
      * task.jsx
  + index.css
  + index.js
  + reportWebVitals.js
  + setupTests.js
* index.json
* package.json
* package-lock.json
* README.md

Файл index.js в корне приложения

const express = require('express')

const userRouter = require('./src/app/router/userRoutes')

const PORT = process.env.PORT || 8080

const app = express()

var cors = require('cors')

app.use(cors())

app.use(express.json())

app.use('/api', userRouter)

app.listen(PORT, () => console.log('server started on port ' + PORT))

Директория src:

Файл index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './app/App';

import { Provider } from "react-redux";

import store from './app/store/store'

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<Provider store={store}>

<App />

</Provider>

);Файл index.css

body {

margin: 0;

font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

code {

font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

monospace;

}

Директория app:

Файл App.js

import { BrowserRouter } from "react-router-dom";

import 'bulma/css/bulma.min.css';

import Router from './Router';

import { useSelector } from 'react-redux';

import './App.scss';

import cn from "classnames";

const App = () => {

return (

<BrowserRouter>

<div className="upsilon-app">

<div className="upsilon">

<Router />

</div>

</div>

</BrowserRouter>

);

}

export default App;

Файл App.scss:

@import url('https://fonts.googleapis.com/css2?family=Ubuntu:wght@300;400;700&display=swap');

.upsilon-app {

display: flex;

min-height: 100vh;

}

body{

background-color: whitesmoke;

}

Файл costants.js:

const APP\_NAME = 'Название';

export {

APP\_NAME

}

Файл db.js:

const Pool = require('pg').Pool

const pool = new Pool({

user: 'postgres',

host: '46.146.117.120',

database: 'postgres',

password: '617',

port: 5432,

});

module.exports = pool

Файл Router.js:

import {

Routes,

Route,

} from "react-router-dom";

import LoginPage from '../pages/Login';

import RegisterUser from '../pages/RegisterUser';

import RegisterEmployer from '../pages/RegisterEmployer';

import Task from '../pages/Task';

import Rezume from '../pages/Rezume';

import NotFound from "../pages/NotFound";

const Router = () => (

<Routes>

<Route path="/" element={<Task />} />

<Route path="/rezume" element={<Rezume />} />

<Route path="/registerUser" element={<RegisterUser />} />

<Route path="/login" element={<LoginPage />} />

<Route path="/registerEmployer" element={<RegisterEmployer />} />

<Route path="/notfound" element={<NotFound />} />

</Routes>

);

export default Router;

Директория components:

Директория Header:

Файл index.jsx:

import { Navbar, Heading } from 'react-bulma-components';

import './Header.scss';

import { useSelector, useDispatch } from "react-redux";

import React from 'react';

import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';

import {

faBars,

faCircleUser,

faArrowRightFromBracket

} from '@fortawesome/free-solid-svg-icons';

import { Link } from 'react-router-dom';

const Header = () => {

return (

<Navbar className='header'>

<div className='icon header-bars' >

<FontAwesomeIcon

icon={ faBars }

size='lg' />

</div>

<div className='header-logo'>Upsilon</div>

<div className='header-accountInfo'>

<Link className='icon accountInfo-icon' to={'/account'}>

<FontAwesomeIcon

icon={ faCircleUser }

size='2x' />

</Link>

<div className='accountInfo-name'></div>

<Link className='icon accountInfo-exit' to={'/'}>

<FontAwesomeIcon

icon={ faArrowRightFromBracket }

size='lg' />

</Link>

</div>

</Navbar>

)

}

export default Header;

Файл Header.scss:

.header {

height: 47px;

display: flex;

justify-content: normal;

align-items: center;

width: 100%;

border-bottom: 1px solid lightgray;

}

.header-bars {

padding-left: 20px;

margin-bottom: 0px !important;

width: 8%;

cursor: pointer;

}

.header-logo {

font-size: 24px;

margin-bottom: 0px !important;

width: 77%;

text-align: center;

}

.header-accountInfo {

display: flex;

height: inherit;

align-items: center;

margin-right: 20px;

}

.accountInfo-icon {

margin-right: 7px;

text-decoration: none;

color: black;

}

.accountInfo-name {

font-size: 16px;

margin-right: 20px;

}

.accountInfo-exit {

color: #3C51E0;

}

Директория reportGenerator:

Файл reportGenerator.js:

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;

Директория Rezume:

Файл index.jsx:

import React, { useState } from 'react';

import './Rezume.scss';

function RezumeCard(props) {

const { education, faculty, description, seekername, seekersurname} = props.rezume;

return (

<section className='hero has-background-grey-lighter is-fullheight section-auth'>

<form className='box box-rezume'>

<div>

<div class="con4">

<img className="img" type="userFoto" alt="Аватарка" scr="..."></img>

{/\* <img >{s(value).userFoto}</img> \*/}

</div>

</div>

<div className='field-rezume '>

<div class="con1">

<output class="output" type="user">{seekername} {seekersurname}</output>

{/\* <pre >{s(value).user}</pre> \*/}

</div>

</div>

<div className='field-rezume'>

<div class="con2">

<output class="output" type="faculty">{faculty}</output>

{/\* <pre >{s(value).faculty}</pre> \*/}

</div>

</div>

<div className='field-rezume'>

<div class="con3">

<output class="output" type="ed\_name">{education}</output>

{/\* <pre >{s(value).ed\_name}</pre> \*/}

</div>

</div>

<div className='field-rezume'>

<div class="con">

<output class="output" type="description">{description}</output>

{/\* <pre >{s(value).description}</pre> \*/}

</div>

</div>

<div className='counts'></div>

</form>

</section>

)

}

export default RezumeCard;

Файл Rezume.scss:

.counts {

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

font-weight: bold !important;

background-color: #89878d !important;

color: rgb(252, 249, 249) !important;

font-size: 20px !important;

position: right !important;

text-align: center;

width: 200px !important;

height: 35px !important;

border-radius: 10px !important;

left: 300px;

}

/\* Box \*/

.box-rezume {

background-color: white !important;

width: 580px !important;

height: 360px !important;

margin: 0 auto !important;

margin-top: 66px !important;

margin-bottom: 10px !important;

padding: 50px 47px !important;

border-radius: 20px !important;

}

.field-rezume {

margin-bottom: 35px !important;

height: 50px !important;

border-radius: 20px !important;

}

.con

{

height: 50px ;

background-color: #d8d8da ;

font-size: 20px ;

border-radius: 5px ;

}

.con1

{

font-weight: bold;

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

font-size: 24px;

color: #000;

}

.con2

{

font-weight: bold;

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

color: #444343;

}

.con3

{

font-weight: bold;

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

color: #444343;

}

Директория Task:

Файл index.jsx:

import React from 'react';

import './taskCard.scss'

function TaskCard(props) {

const { employer, name, faculty, description, workername, workersurname, price, education, phone, date} = props.task;

return (

<div className='box box-card'>

<div>

</div>

<div className='field'>

<div className="field-employer">

<output className="output" type="user">{employer}</output>

{/\* <pre >{s(value).user}</pre> \*/}

</div>

</div>

<div className='field'>

<div className="field-name">

<output className="output" type="title">{name}</output>

{/\* <pre >{s(value).title}</pre> \*/}

</div>

</div>

<div className='field'>

<div className="field-faculty">

<output className="output" type="faculty">{faculty}</output>

{/\* <pre >{s(value).faculty}</pre> \*/}

</div>

</div>

<div className='field'>

<div className="field-description">

<output className="output" type="description" >{description}...</output>

{/\* <pre >{s(value).description}</pre> \*/}

</div>

</div>

<div className='field'>

<div className="field-worker">

<output className="output" type="ed\_name">Консультант {workername} {workersurname}</output>

{/\* <pre >{s(value).ed\_name}</pre> \*/}

</div>

</div>

</div>

)

}

export default TaskCard;

Файл taskCard.scss:

.has-background-grey-lighter

{

}

/\* Box \*/

.box-card

{

margin: 0 0 35px 250px;

border-radius: 25px !important;

width:935px ;

}

.field

{

letter-spacing: 1px;

}

.field-img

{

justify-content:right;

}

.field-employer

{

text-transform: uppercase;

font-size: 16px;

color: #6E7191;

font-weight: bold;

letter-spacing: 3px;

}

.field-name

{

width: 850px;

font-weight: bold;

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

font-size: 24px;

color: #14142B;

letter-spacing: 0px;

}

.field-faculty

{

text-align: center;

width: 500px;

height: 30px;

font-weight: bold;

font-size: 18px;

background-color: #F0F1F5 ;

border-radius: 5px ;

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

color: #9C9DB5;

}

.field-description

{

width: 850px;

font-size: 16px;

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

color: #14142B;

}

.field-worker

{

position:static;

left: 100px;

top: 450px;

text-transform: uppercase;

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

color: #A0A3BD;

font-weight: bold;

letter-spacing: 3px;

margin-bottom: 15px;

}

Директория controller:

Файл rezumeController.js:

const db = require('../db')

class RezumeController {

async createRezume(req, res){

const {name, surname} = req.body

const newPerson = await db.query('INSERT INTO public.task (u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo) values ($1, $2, $3, $4, $5, $6, $7, $8) RETURNING \*', [u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo])

res.json(newPerson)

}

async getRezume(req, res){

const rezumes = await db.query('select u\_name as seekername, f\_name as faculty, ed\_name as education, s\_description as description, u\_surname as seerkerSurnamefrom faculty, seeker, education, public.user where (seeker.faculty\_idfaculty = f\_id) and (seeker.education\_ideducation=ed\_id) and (seeker.user\_iduser=public.user.id)')

res.json(rezumes.rows)

}

}

module.exports = new RezumeController()

Файл taskController.js:

const db = require('../db')

class TaskController {

async createTask(req, res){

const {name, surname} = req.body

const newPerson = await db.query('INSERT INTO public.task (u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo) values ($1, $2, $3, $4, $5, $6, $7, $8) RETURNING \*', [u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo])

res.json(newPerson)

}

async getTask (req, res){

const tasks = await db.query(`select task.id as id,

em\_name as employer,

t\_name as name, f\_name as faculty,

SUBSTRING(t\_description, 1, 200) AS description,

t\_price as price,

u\_name as workerName,

u\_surname as workerSurname,

ed\_name as education,

t\_date as date

from faculty, task, employer, worker, public.user, education

where (task.faculty\_idfaculty = f\_id)

and (task.employer\_idemployer=em\_id)

and (task.worker\_idWorker = w\_id)

and (worker.user\_iduser=public.user.id)

AND task.education\_ideducation = education.ed\_id`)

res.json(tasks.rows)

}

async getEmployers(req, res){

const empoyers = await db.query("SELECT em\_id, em\_name from employer")

res.json(empoyers.rows)

}

async getFaculties(req, res){

const empoyers = await db.query("SELECT f\_id, f\_name from faculty")

res.json(empoyers.rows)

}

async getFilteredTask(req, res){

const employer\_ids = req.body.employer\_ids;

const faculty\_ids = req.body.faculty\_ids;

//const employer\_ids = employer\_ids\_param ? employer\_ids\_param.split(',') : [];

//const faculty\_ids = faculty\_ids\_param ? faculty\_ids\_param.split(',') : [];

let query = `

SELECT task.id as id,

em\_name AS employer,

t\_name AS name,

f\_name AS faculty,

SUBSTRING(t\_description, 1, 200) AS description,

t\_price AS price,

u\_name AS workerName,

u\_surname AS workerSurname,

ed\_name as education,

t\_date as date

FROM faculty, task, employer, worker, public.user, education

WHERE task.faculty\_idfaculty = f\_id

AND task.employer\_idemployer = em\_id

AND task.worker\_idWorker = w\_id

AND task.education\_ideducation = education.ed\_id

AND worker.user\_iduser = public.user.id

`;

if (employer\_ids.length > 0) {

query += `AND employer.em\_id IN (${employer\_ids.join(',')}) `;

}

if (faculty\_ids.length > 0) {

query += `AND faculty.f\_id IN (${faculty\_ids.join(',')}) `;

}

try {

const tasks = await db.query(query);

res.json(tasks.rows);

} catch (error) {

console.error(error);

res.status(500).send('Internal Server Error');

}

}

async getIdTask(req, res){

const id = req.params.id

const task = await db.query('SELECT \* FROM public.task where id = $1', [id])

res.json(task.rows[0])

}

async updateTask(req, res){

const {u\_id, u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo} = req.body

const user = await db.query('UPDATE public.task set u\_login = $1, u\_password = $2, u\_surname = $3, u\_name = $4, u\_patronymic = $5, u\_email = $6, u\_level = $7, u\_photo = $8 where u\_id = $9 RETURNING \*', [u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo, u\_id])

res.json(user.rows[0])

}

async deleteTask(req, res) {

const id = req.params.id

const users = await db.query('DELETE FROM public.task where u\_id = $1', [id])

res.json(users.rows[0])

}

async authUser(req, res) {

const { username, password } = req.query;

const users = await db.query('SELECT \* FROM public.task where ((u\_login = $1 and u\_password = $2) or (u\_email = $1 and u\_password = $2))', [username, password])

//console.log(username)

//console.log(password)

//console.log('auth try')

res.json(users.rows[0])

}

}

module.exports = new TaskController()

Файл usersController.js:

const db = require('../db')

class UserController {

async createUser(req, res){

const {name, surname} = req.body

const newPerson = await db.query('INSERT INTO public.user (u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo) values ($1, $2, $3, $4, $5, $6, $7, $8) RETURNING \*', [u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo])

res.json(newPerson)

}

async getUsers (req, res){

const users = await db.query('SELECT \* FROM public.user')

res.json(users.row)

}

async getOneUser(req, res){

const id = req.params.id

const users = await db.query('SELECT \* FROM public.user where u\_id = $1', [id])

res.json(users.rows[0])

}

async updateUser(req, res){

const {u\_id, u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo} = req.body

const user = await db.query('UPDATE public.user set u\_login = $1, u\_password = $2, u\_surname = $3, u\_name = $4, u\_patronymic = $5, u\_email = $6, u\_level = $7, u\_photo = $8 where u\_id = $9 RETURNING \*', [u\_login, u\_password, u\_surname, u\_name, u\_patronymic, u\_email, u\_level, u\_photo, u\_id])

res.json(user.rows[0])

}

async deleteUser(req, res) {

const id = req.params.id

const users = await db.query('DELETE FROM public.user where u\_id = $1', [id])

res.json(users.rows[0])

}

async authUser(req, res) {

const { username, password } = req.query;

const users = await db.query('SELECT \* FROM public.user where ((u\_login = $1 and u\_password = $2) or (u\_email = $1 and u\_password = $2))', [username, password])

//console.log(username)

//console.log(password)

//console.log('auth try')

res.json(users.rows[0])

}

}

module.exports = new UserController()

Директория router:

Файл iserRouter.js:

const Router = require('express')

const router = new Router()

const userController = require('../controller/userController')

const taskController = require('../controller/taskController')

const rezumeController = require('../controller/rezumeController')

router.post('/user', userController.createUser)

router.get('/user', userController.getUsers)

router.post('/user/auth', userController.authUser)

router.post('/task', taskController.getTask)

router.post('/task/filter', taskController.getFilteredTask)

router.post('/rezume', rezumeController.getRezume)

router.get('/employer', taskController.getEmployers)

router.get('/faculty', taskController.getFaculties)

module.exports = router

Директория store:

Файл store.js:

import { configureStore } from '@reduxjs/toolkit';

export default configureStore({

devTools: process.env.NODE\_ENV !== 'production',

reducer:{

}

});

Директория pages:

Директория Login:

Файл index.jsx:

import React, { useState } from 'react';

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

import { useNavigate } from 'react-router-dom';

import { useDispatch } from 'react-redux';

import './LoginPage.scss';

import \* as Yup from 'yup';

import { useForm } from "react-hook-form";

import { yupResolver } from "@hookform/resolvers/yup";

const schema = Yup.object().shape({

username: Yup

.string()

.required('Поле не может быть пустым'),

password: Yup

.string()

.required('Поле не может быть пустым')

.min(6, 'Поле должно содержать не менее 6 символов')

});

const LoginPage = () => {

/\*const { register, handleLogin, formState: { errors } } = useForm({

mode: "onSubmit",

resolver: yupResolver(schema)

})\*/

return (

<section className='hero has-background-grey-lighter is-fullheight section-auth-login'>

<form className='box box-login' /\*onSubmit={handleLogin}\*/>

<div className='field field-login'>

<div className="control">

<input

className="input"

type="text"

id="username"

placeholder="Логин/E-mail"

/\*value = {username}

/\*onChange={(e) => setUsername(e.target.value)}{...("username")}\*//>

</div>

</div>

<div className='field field-login'>

<div class="control">

<input

className="input"

type="password"

id="password"

placeholder="\*\*\*\*\*\*\*\*\*\*\*\*"

/\*value = {password}

/\*onChange={(e) => setPassword(e.target.value)}{...("password")}\*//>

</div>

</div>

<div className='field'>

<Button

className='button is-fullwidth button-login'

type='submit'>

Войти

</Button>

</div>

</form>

<form className='box box-registration-login' /\*onSubmit={handleLogin(onSubmit)}\*/>

<div className='field'>

<Button

className='button is-fullwidth button-registration'

type='submit'>

Зарегитрироваться

</Button>

</div>

<div className='field'>

<p className='registration-info'>После регистрации вы получите доступ к размещению резюме</p>

</div>

</form>

</section>

)

}

export default LoginPage;

Файл LoginPage.scss:

.upsilon-login-page {

width: 100vw;

height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

background-color: #F5F5F9;

}

.section-auth {

width: 100vw;

}

.app-name {

text-align: center;

margin-top: 29px;

font-weight: bold;

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

font-size: 24px;

color: #000;

}

.button-login {

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

font-weight: bold !important;

background-color: #7C41DE !important;

color: white !important;

font-size: 24px !important;

text-align: center;

width: 492px !important;

height: 60px !important;

border-radius: 15px !important;

}

.button-registration {

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

font-weight: bold !important;

background-color: #7C41DE !important;

color: white !important;

font-size: 24px !important;

text-align: center;

width: 492px !important;

height: 60px !important;

border-radius: 15px !important;

}

.field-button{

display: flex;

justify-content: space-around;

}

/\* Box \*/

.box-login {

background-color: white !important;

width: 580px !important;

height: 350px !important;

margin: 0 auto !important;

margin-top: 66px !important;

margin-bottom: 10px !important;

padding: 50px 47px !important;

border-radius: 20px !important;

}

.field-login {

margin-bottom: 35px !important;

height: 50px !important;

border-radius: 20px !important;

}

.input

{

height: 50px !important;

border-radius: 15px !important;

}

.registration-info {

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

color: #6e6e6e ;

font-size: 20px;

font-weight: lighter;

text-align: center;

}

.box-registration-login {

background-color: white !important;

width: 580px !important;

height: 258px !important;

margin: 0 auto !important;

margin-top: 10px !important;

margin-bottom: 600px !important;

padding: 78px 47px !important;

border-radius: 20px !important;

}

Директория NotFound:

Файл index.jsx:

import {

Box,

Button,

} from 'react-bulma-components';

import './NotFound.scss';

const NotFound = () => {

return (

<div className="mon-notfound">

<Box>

<div className={"NotF-head"}>

<p class="title is-1 ">

Упс... 404 </p>

<p class="subtitle is-3">Страница не найдена</p>

</div>

<div className="NotF-middle">

<p>Этому может быть несколько объяснений:</p>

<ul >

<li>Страница удалена или перемещена</li>

<li>Вы ошиблись при вводе адреса страницы</li>

</ul>

</div>

</Box>

</div>

)

}

export default NotFound;

Файл NotFound.scss:

// Google Fonts

@import url('https://fonts.googleapis.com/css2?family=Roboto&family=Rubik+Mono+One&display=swap');

// Font Variables

$rubik-head: 'Rubik Mono One', sans-serif;

$roboto-content: 'Roboto', sans-serif;

.mon-notfound {

margin: 200px 600px 0 700px;

width: auto;

height: calc(100vh - 54px);

display: flex;

flex-direction: column;

//background-color: #F5F5F9;

padding: 24px;

.box{

padding: 3rem;

margin: 0;

align-items: center;

font-family: $roboto-content;

font-size: 1em;

.NotF-head {

font-family: $rubik-head;

.title{

color: rgba(105, 104, 104, 0.6);

}

.subtitle{

color: rgba(105, 104, 104, 0.3);

}

}

.NotF-middle {

padding-top: 2rem;

}

.NotF-bottom {

}

}

}

ul {

display: block;

list-style-type: square;

padding-inline-start: 1em;

padding-block-start: 0.5em;

padding-block-end: 1em;

}

Директория RegisterEmployer:

Файл index.jsx:

import React, { useState } from 'react';

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

import './RegisterPageEmployer.scss';

const RegisterPage = () => {

return (

<section className='hero has-background-grey-lighter is-fullheight section-auth-emp'>

<form className='box box-register-emp' /\*onSubmit={handleSubmit(onSubmit)}\*/>

<div className='field field-button'>

<div className='field'>

<Button

className='button is-fullwidth button-seeker-emp'

type='submit'>

Соискатель

</Button>

</div>

<div className='field'>

<Button

className='button is-fullwidth button-employer-emp'

type='submit'>

Работодатель

</Button>

</div>

</div>

<h2>Регистрация организации </h2>

<div className='field field-register-emp'>

<h4>Название организации</h4>

<div className="control">

<input className="input" type="email" placeholder="Введите e-mail"></input>

</div>

</div>

<div className='field field-register-emp'>

<h4>Описание</h4>

<div className="control">

<input className="input" type="password" placeholder="Введите пароль"></input>

</div>

</div>

<h2>Регистрация сотрудника</h2>

<div className='field field-register-emp'>

<h4>Имя</h4>

<div className="control">

<input className="input" type="name" placeholder="Ваше Имя"></input>

</div>

</div>

<div className='field field-register-emp'>

<h4>Фамилия</h4>

<div className="control">

<input className="input" type="surname" placeholder="Ваша фамилия"></input>

</div>

</div>

<div className='field field-register-emp'>

<h4>Отчество</h4>

<div class="control">

<input class="input" type="patronimic" placeholder="Ваше Отчество"></input>

</div>

</div>

<div className='field field-register-emp'>

<h4>Email</h4>

<div className="control">

<input className="input" type="email" placeholder="Введите e-mail"></input>

</div>

</div>

<div className='field field-register-emp'>

<h4>Контактный телефон</h4>

<div className="control">

<input className="input" type="phone-number" placeholder="Введите телефон"></input>

</div>

</div>

<div className='field field-register-emp'>

<h4>Пароль</h4>

<div className="control">

<input className="input" type="password" placeholder="Введите пароль"></input>

</div>

</div>

<div className='field field-register-emp'>

<div className="control">

<input className="input" type="confirm-password" placeholder="Повторите пароль"></input>

</div>

</div>

<div className='field'>

<Button

className='button is-fullwidth button-registration'

type='submit'>

Зарегитрироваться

</Button>

</div>

</form>

</section>

)

}

export default RegisterPage;

Файл RegisterPageEmployer.scss:

.upsilon-login-page {

width: 100%;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

background-color: #2d2d97;

}

.section-auth-emp {

width: 100vw;

}

.app-name {

text-align: center;

margin-top: 29px;

font-weight: bold;

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

font-size: 24px;

color: #000;

}

.button-login {

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

font-weight: bold !important;

background-color: #7C41DE !important;

color: white !important;

font-size: 24px !important;

text-align: center;

width: 492px !important;

height: 51px !important;

border-radius: 15px !important;

}

.button-registration {

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

font-weight: bold !important;

background-color: #7C41DE !important;

color: white !important;

font-size: 24px !important;

text-align: center;

width: 492px !important;

height: 51px !important;

border-radius: 15px !important;

}

.field-button{

display: flex;

justify-content: space-around;

}

.button-employer-emp{

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

font-weight: bold !important;

background-color: #7C41DE !important;

color: white !important;

font-size: 24px !important;

text-align: center;

width: 241px !important;

height: 63px !important;

border-radius: 15px !important;

margin-left: 5px;

}

.button-seeker-emp{

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

font-weight: bold !important;

background-color: #F0F1F5 !important;

color: #9C9DB5 !important;

font-size: 24px !important;

text-align: center;

width: 241px !important;

height: 63px !important;

border-radius: 15px !important;

margin-left: 5px;

}

/\* Box \*/

.box-register-emp {

background-color: white !important;

width: 580px !important;

height: 1250px !important;

margin: 0 auto !important;

margin-top: 66px !important;

margin-bottom: 10px !important;

padding: 50px 47px !important;

border-radius: 20px !important;

}

.field-register-emp{

margin-bottom: 35px !important;

margin-top: 15px!important;

height: 70px !important;

}

.registration-info {

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

color: #6e6e6e ;

font-size: 20px;

font-weight: lighter;

text-align: center;

}

.box-registration-emp {

background-color: white !important;

width: 580px !important;

height: 258px !important;

margin: 0 auto !important;

margin-top: 10px !important;

margin-bottom: 311px !important;

padding: 78px 47px !important;

border-radius: 20px !important;

}

h2

{

text-align: center;

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

font-weight: bold !important;

font-size: large;

color: #000;

}

h2

{

text-align: center;

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

font-weight: bold !important;

color: #000;

}

.box h4

{

justify-content: center;

}

input::-webkit-input-placeholder { color: #6E6E6E!important; }

Директория RegisterUser:

Файл index.jsx:

import React, { useState } from 'react';

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

import './RegisterPageUser.scss';

import Task from "../../app/components/Task";

const RegisterPage = () => {

return (

<section className='hero has-background-grey-lighter is-fullheight section-auth-user'>

<form className='box box-register-user' /\*onSubmit={handleSubmit(onSubmit)}\*/>

<div className='field field-button'>

<div className='field'>

<Button

className='button is-fullwidth button-seeker-user'

type='submit'>

Соискатель

</Button>

</div>

<div className='field'>

<Button

className='button is-fullwidth button-employer-user'

type='submit'>

Работодатель

</Button>

</div>

</div>

<div className='field field-register-user'>

<div className="control">

<input className="input" type="email" placeholder="Введите e-mail"></input>

</div>

</div>

<div className='field field-register-user'>

<div className="control">

<input className="input" type="password" placeholder="Введите пароль"></input>

</div>

</div>

<div className='field field-register-user'>

<div className="control">

<input className="input" type="confirm-password" placeholder="Повторите пароль"></input>

</div>

</div>

<div className='field field-register-user'>

<div className="control">

<input className="input" type="name" placeholder="Ваше Имя"></input>

</div>

</div>

<div className='field field-register-user'>

<div className="control">

<input className="input" type="surname" placeholder="Ваша фамилия"></input>

</div>

</div>

<div className='field'>

<Button

className='button is-fullwidth button-registration'

type='submit'>

Зарегитрироваться

</Button>

</div>

</form>

</section>

)

}

export default RegisterPage;

Файл RegisterPageUser.scss:

.upsilon-login-page {

width: 100%;

height: calc(100vh - 54px);

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

background-color: #F5F5F9;

}

.section-auth-user {

width: 100vw;

}

.app-name {

text-align: center;

margin-top: 29px;

font-weight: bold;

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

font-size: 24px;

color: #000;

}

.button-login {

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

font-weight: bold !important;

background-color: #7C41DE !important;

color: white !important;

font-size: 24px !important;

text-align: center;

width: 492px !important;

height: 51px !important;

border-radius: 15px !important;

}

.button-registration {

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

font-weight: bold !important;

background-color: #7C41DE !important;

color: white !important;

font-size: 24px !important;

text-align: center;

width: 492px !important;

height: 51px !important;

border-radius: 15px !important;

}

.field-button{

display: flex;

justify-content: space-around;

}

.button-employer-user{

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

font-weight: bold !important;

background-color: #F0F1F5 !important;

color: #9C9DB5 !important;

font-size: 24px !important;

text-align: center;

width: 241px !important;

height: 63px !important;

border-radius: 15px !important;

margin-left: 5px;

}

.button-seeker-user{

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

font-weight: bold !important;

background-color: #7C41DE !important;

color: white !important;

font-size: 24px !important;

text-align: center;

width: 241px !important;

height: 63px !important;

border-radius: 15px !important;

margin-right: 5px;

}

/\* Box \*/

.box-register-user {

background-color: white !important;

width: 580px !important;

height: 670px !important;

margin: 0 auto !important;

margin-top: 66px !important;

margin-bottom: 10px !important;

padding: 50px 47px !important;

border-radius: 20px !important;

}

.field-register-user {

margin-bottom: 15px !important;

height: 70px !important;

}

input::-webkit-input-placeholder { color: #6E6E6E!important; }

.registration-info {

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

color: #6E6E6E ;

font-size: 20px;

font-weight: lighter;

text-align: center;

}

.box-registration-user {

background-color: white !important;

width: 580px !important;

height: 400px !important;

margin: 0 auto !important;

margin-top: 10px !important;

margin-bottom: 311px !important;

padding: 78px 47px !important;

border-radius: 20px !important;

}

Директория Rezume:

Файл index.jsx:

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

import axios from 'axios';

import RezumeCard from '../../app/components/Rezume';

function Rezume() {

const [rezumes, setRezumes] = useState([]);

useEffect(() => {

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

.then(response => {

setRezumes(response.data);

console.log(response.data)

})

.catch(error => {

console.log(error);

});

}, []);

return (

<div>

<h2>Rezume</h2>

{Array.from(rezumes).map(rezume => (

<RezumeCard key={rezume.id} rezume={rezume}/>

))}

</div>

);

}

export default Rezume;

Директория Rezume:

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;

}