Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Южно-Уральский государственный университет»

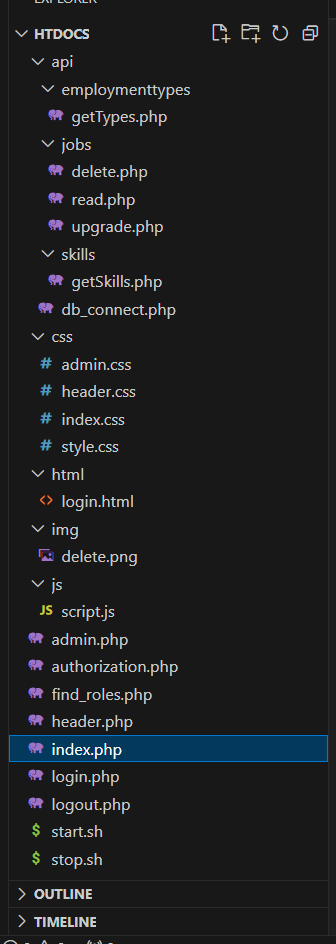
(национальный исследовательский университет)

Институт естественных и точных наук

Кафедра прикладной математики и программирования

Отчёт о выполнении индивидуального проекта «Учёт сотрудников» по дисциплине «Web-программирование»

|  |  |
| --- | --- |
|  | Руководитель работы,  \_\_\_\_\_\_\_\_\_\_/Бунова Е.В.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г. |
|  | Автор работы  Студент группы ЕТ-413  \_\_\_\_\_\_\_\_\_\_\_\_\_/Абзалимов К.В.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г. |
|  |  |

Физическая структура проекта  


База данных

## Таблица Роли

Название таблицы: roles.

Таблица 1.1 – Роли. Структура

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **NOT NULL** | **Примечание** |
| id | serial | Да | Первичный ключ |
| code | character varying(20) | Да | Уникальность на (code) |
| name | character varying(40) | Да | Уникальность на (name) |

## Таблица Пользователей

Название таблицы: users.

Таблица 1.2 – Пользователи. Структура

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **NOT NULL** | **Примечание** |
| id | serial | Да | Первичный ключ |
| name | character varying(40) | Да |  |
| login | character varying(10) | Да | Уникальность на (login) |
| pass | character varying(20) | Да |  |

## Таблица Роли пользователей

Название таблицы: user\_roles.

Таблица 1.3 – Роли пользователей. Структура

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **NOT NULL** | **Примечание** |
| id | serial | Да | Первичный ключ |
| role\_id | integer | Да | Внешний ключ на role.id |
| user\_id | integer | Да | Внешний ключ на user.id |

## Таблица типа занятости

Название таблицы employment\_type

Таблица 1.4 – Таблица типов занятости. Структура

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **NOT NULL** | **Примечание** |
| id | serial | Да | Первичный ключ |
| Employment\_name | character varying(50) | Да |  |

## Таблица основного навыка

Название таблицы skills

Таблица 1.4 – Таблица типов занятости. Структура

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **NOT NULL** | **Примечание** |
| id | serial | Да | Первичный ключ |
| Skill\_name | character varying(50) | Да |  |

**Таблица сотрудников**

Название таблицы: jobs.

Таблица 1.5 – Таблица недвижимости. Структура

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип данных | NOT NULL | Примечание |
| id | serial | Да | Первичный ключ |
| employmenttype\_id | integer | Да | Внешний ключ на employment\_type.id |
| Job\_name | double precision | Да |  |
| address | character varying(255) | Да |  |
| salary | integer | Да |  |
| Skill\_id | integer | Да | Внешний ключ на skill.id |

**Страница авторизации**

При открытии страницы authorization.php открывается форма авторизации, в которую необходимо ввести логин и пароль. При вводе существующих в базе данных логина и пароля происходит получение ролей пользователя, которые сохранются в cookies. Происходит переход на страницу index.php

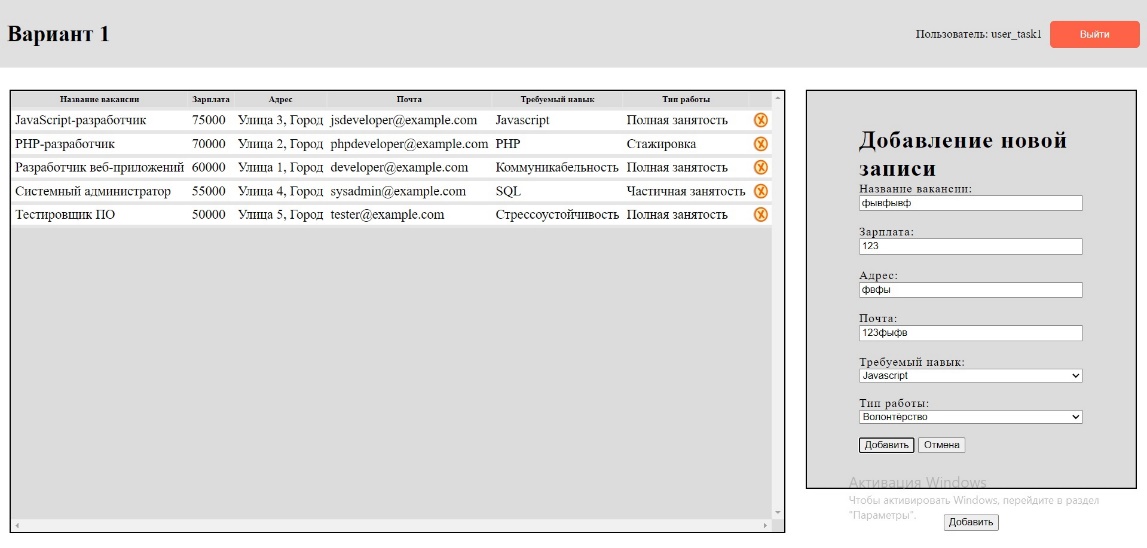


При вводе неверных логина или пароля выводится сообщение об ошибке в форме авторизации

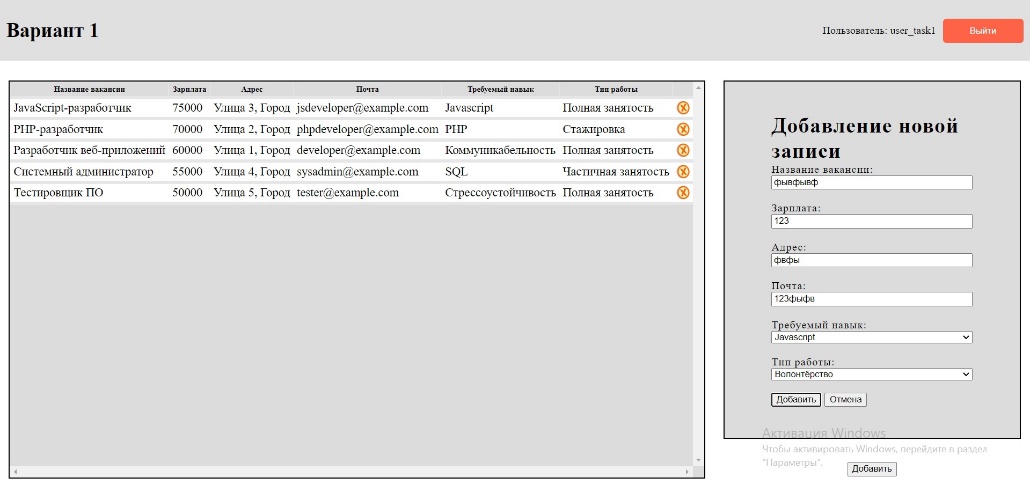


**Страница учёта сотрудников**

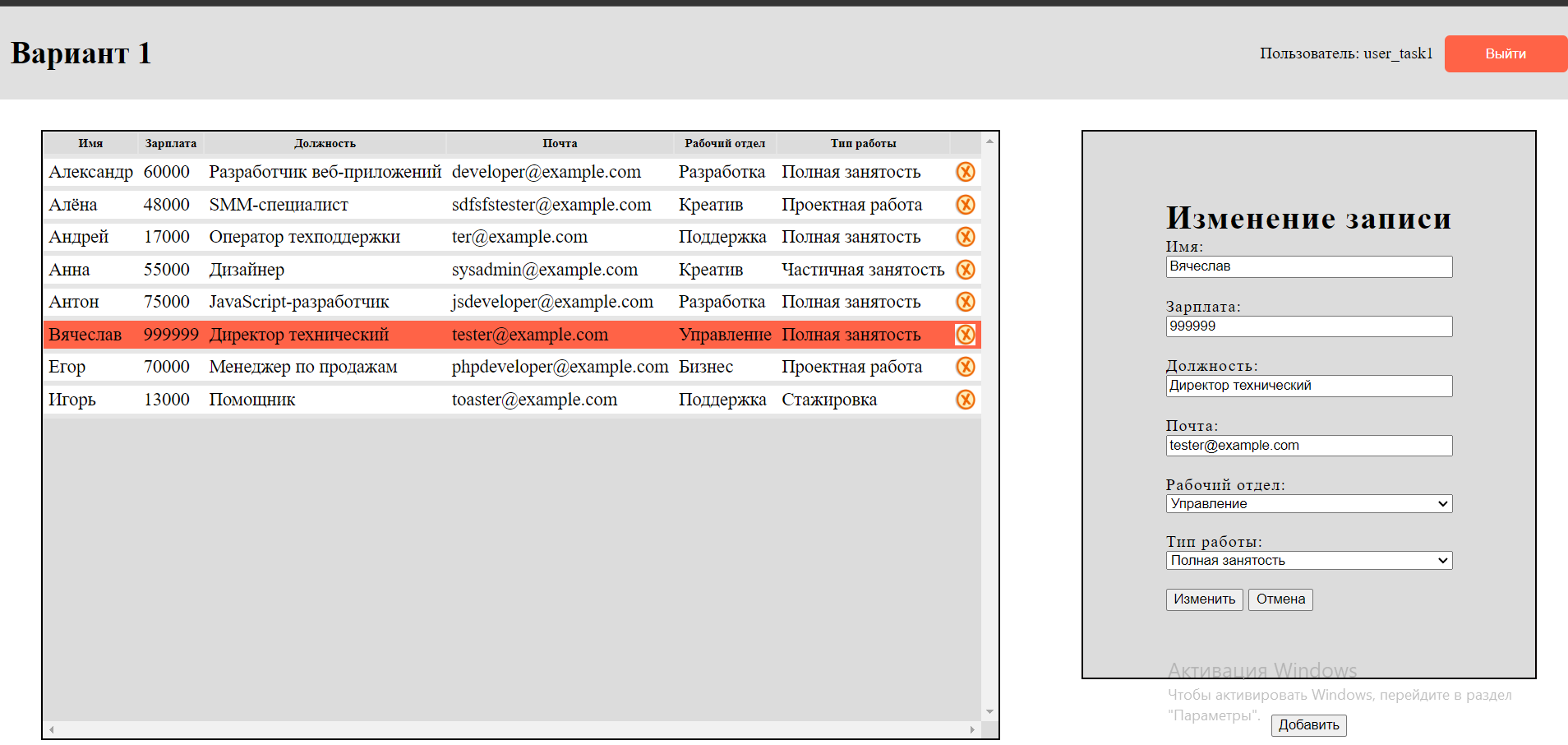
При переходе по ссылке /index.php открывается страница с таблицей недвижимости.



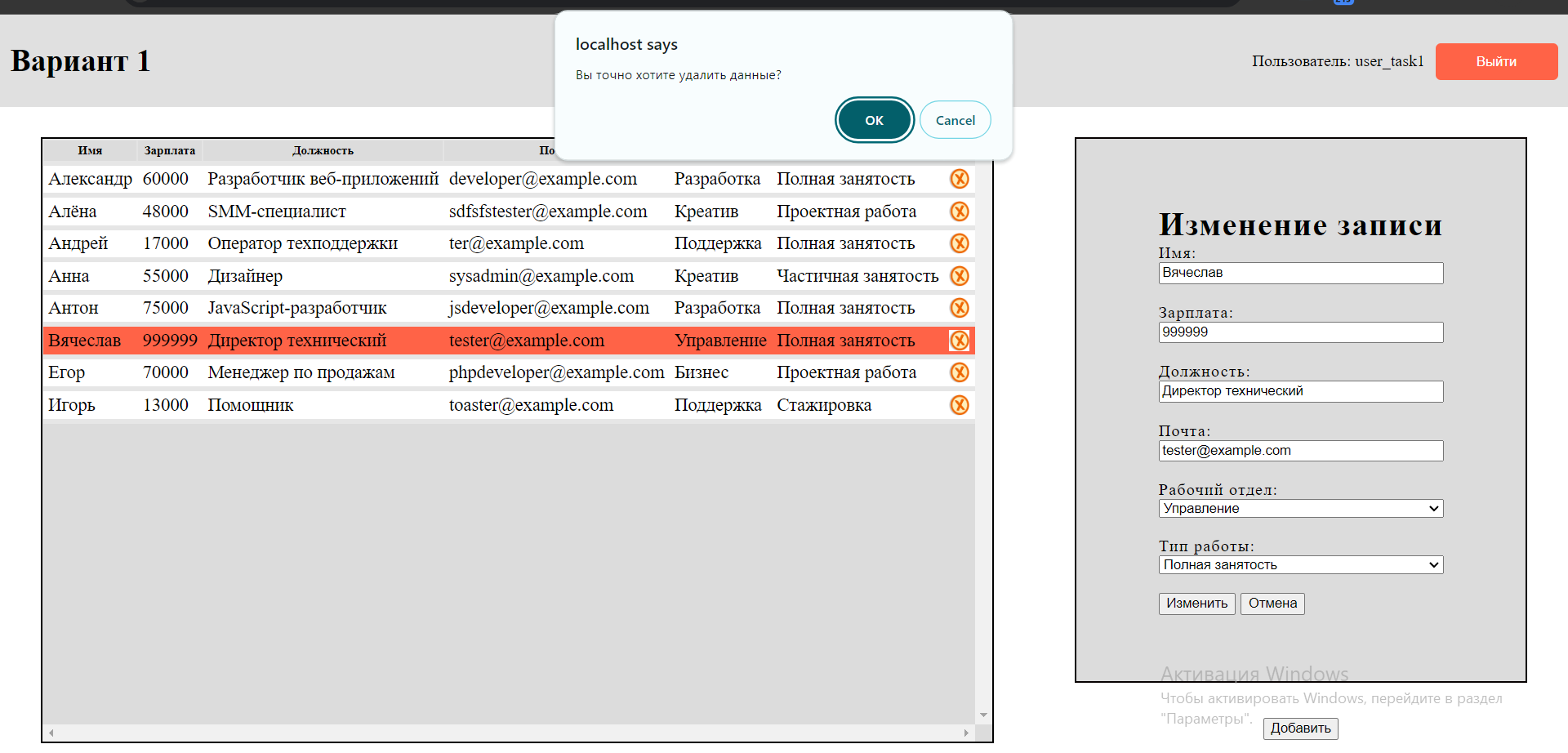
При нажатии на кнопку **Добавить** проявляются поля для добавления в таблицу новой записи. Поля для заполнения: площадь недвижимости, город недвижимости, владелец недвижимости, адрес недвижимости, этаж, на котором располагается недвижимость. При нажатии на кнопку **Отменить** внутри формы поля добавления записи скрываются. При нажатии на кнопку **Добавить** внутри формы происходит валидация всех введенных полей, и в случае некоректно введенных данных появляется сообщение об ошибке. Если данные корректны, новая запись добавляется в базу данных, таблица недвижимости справа обновляется: в нее добавляется новая запись.



При нажатии на запись в таблице недвижимость она становится активной и её можно отредактировать в форме изменения данных слева. При нажатии на кнопку **Отменить** внутри формы поля добавления записи скрываются. При нажатии на кнопку **Изменить** внутри формы происходит валидация всех введенных полей, и в случае некоректно введенных данных появляется сообщение об ошибке. Если данные корректны, запись обновляется в базе данных, таблица недвижимости справа перерисовывается: измененная запись обновляется.

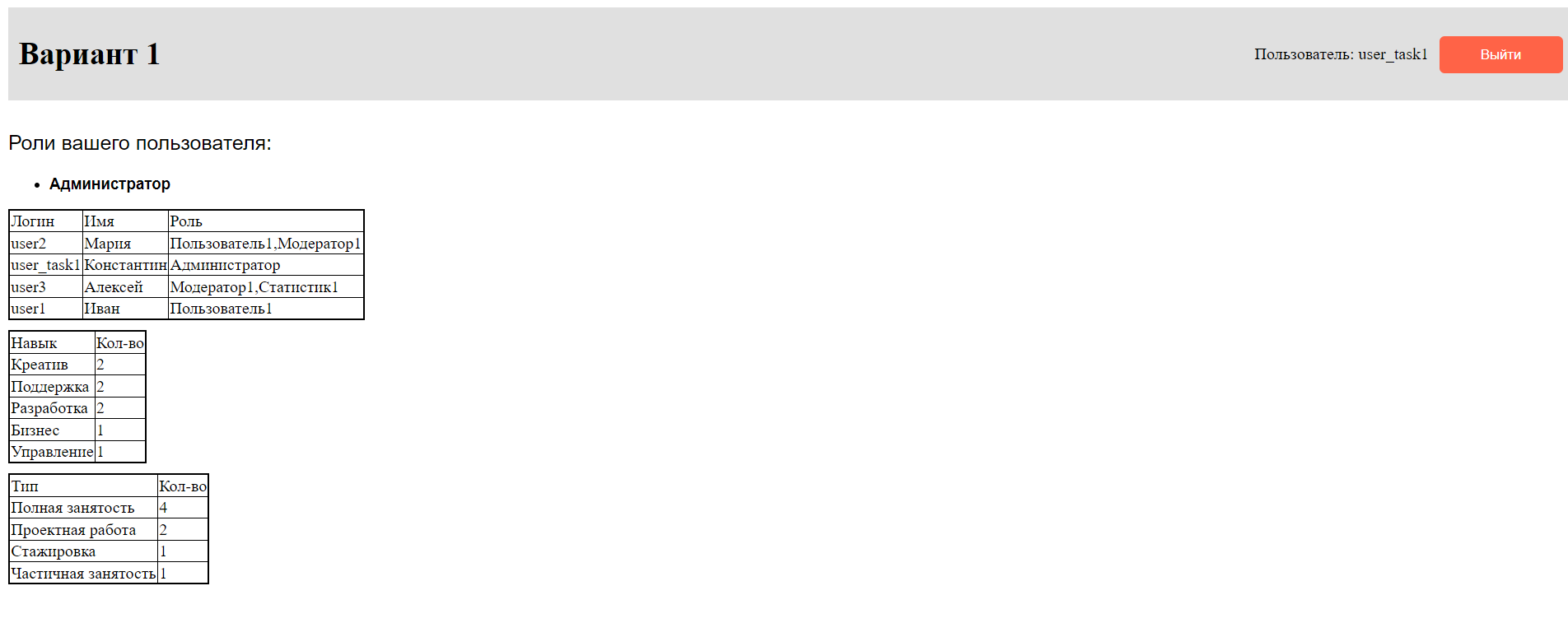


При нажатии на «крестик» на записи в таблице недвижимости, сплывает модальное окно с подтверждением об удалении этой записи. При нажатии на **ОК** запись удаляется в базе данных, таблица недвижимости обновляется: выбранная для удаления запись исчезает.

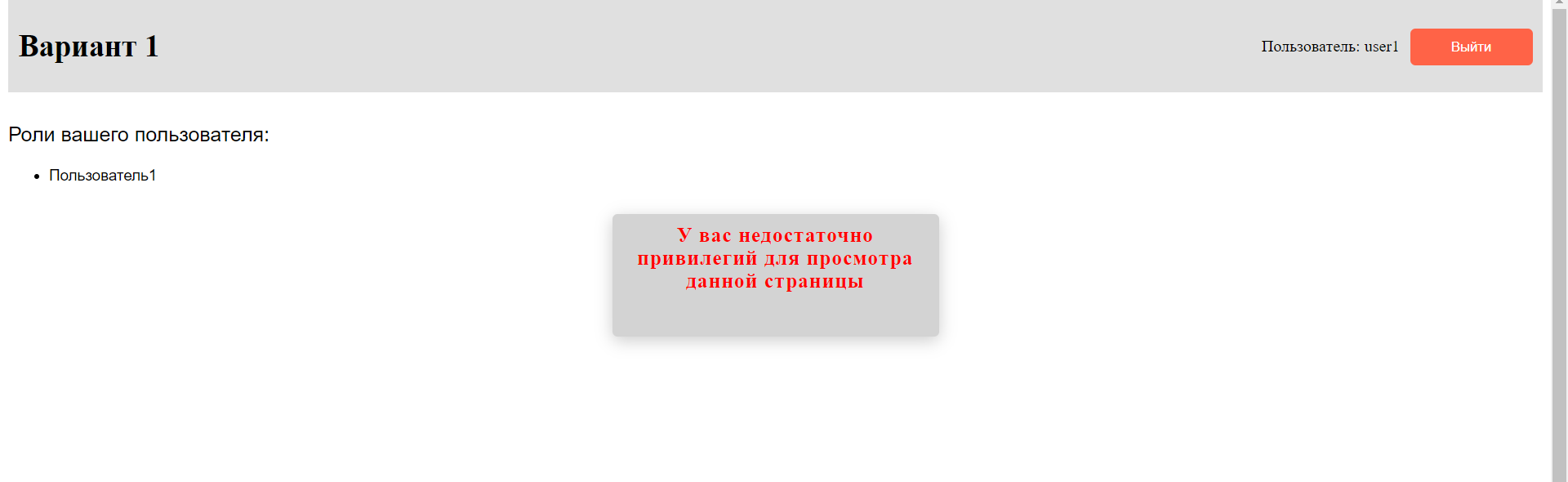


**Страница администратора**

При переходе на страницу /admin.php выводятся 2 таблицы: таблица ролей пользователей и таблица с количеством недвижимости в существующих городах.



Если у пользователя нет нужной роли для доступа к admin.php, то выводить список ролей пользователя и сообщение «У вас недостаточно привилегий для просмотра данной страницы»



Код проекта

Script.js

let mainValues = [];

let activeRowIndex = null;

function userFetch(url, options, callback) {

    fetch(url, options)

    .then(function(response) {

        return response.json();

    })

    .then(function(data) {

        if (data.error == 1) {

            throw(data.errorMsg);

        }

        callback(data.data);

    })

    .catch(function(error) {

        alert(error);

       console.error('Error!!!:', error);

    });

}

userFetch('api/skills/getSkills.php', null, function(data) {

    const rows = [];

    data.forEach(function(item) {

        rows.push(`<option value="${item.skill\_id}">${item.skill\_name}</option>`);

    });

    document.querySelector('#select-skill').innerHTML += rows.join('');

});

userFetch('api/employmenttypes/getTypes.php', null, function(data) {

    const rows = [];

    data.forEach(function(item) {

        rows.push(`<option value="${item.employmenttype\_id}">${item.employmenttype\_name}</option>`);

    });

    document.querySelector('#select-type').innerHTML += rows.join('');

});

updateMain();

function updateMain() {

    setActiveRow(null);

    userFetch('api/jobs/read.php', null, function(data) {

        mainValues = data;

        buildTable();

    });

}

function buildTable() {

    const rows = [];

    mainValues.forEach(function(item, index) {

        const row = `

            <tr data-index="${index}" onclick="rowClick(${index})">

                <td>${item.job\_name}</td>

                <td>${item.salary}</td>

                <td>${item.address}</td>

                <td>${item.email}</td>

                <td>${item.skill\_name}</td>

                <td>${item.employmenttype\_name}</td>

                <td onclick="event.stopPropagation() ; removeRow(${index})">

                    <img width="20px" src="img/delete.png">

                </td>

            </tr>`;

            rows.push(row);

    })

    document.querySelector('#main-container').innerHTML = rows.join('');

}

function setActiveRow(index) {

    if (activeRowIndex !== null) {

        const item = document.querySelector(`[data-index="${activeRowIndex}"]`);

        if (item) {

            item.classList.remove('activeRow');

        }

    }

    activeRowIndex = index;

    if (index === null) {

        ['#job-name', '#job-salary', '#job-address', '#job-email','#select-skill', '#select-type'].forEach(function(id) {

            document.querySelector(id).value = null;

        });

        return;

    }

    const item = document.querySelector(`[data-index="${index}"`);

    if (item) {

        item.classList.add("activeRow");

        document.querySelector('#input-area-title').innerHTML = 'Изменение записи';

        document.querySelector('#input-button').innerHTML = 'Изменить';

        document.querySelector('#input-button').setAttribute('onclick', 'editRow()');

    }

}

function rowClick(index) {

    setActiveRow(index);

    const item = mainValues[index];

    document.querySelector("#job-name").value = item.job\_name;

    document.querySelector("#job-salary").value = item.salary;

    document.querySelector("#job-address").value = item.address;

    document.querySelector("#job-email").value = item.email;

    document.querySelector("#select-skill").value = item.skill\_id;

    document.querySelector("#select-type").value = item.employmenttype\_id;

    document.querySelector('#input-area-title').innerHTML = 'Изменение записи';

    openInputFields();

}

function addRow() {

    const bodyJson = {

        job\_name: document.querySelector('#job-name').value,

        salary: document.querySelector('#job-salary').value,

        address: document.querySelector('#job-address').value,

        email: document.querySelector('#job-email').value,

        skill\_id: document.querySelector('#select-skill').value,

        employmenttype\_id: document.querySelector('#select-type').value

    };

    userFetch(`api/jobs/upgrade.php`, {

        method: 'POST',

        body: JSON.stringify(bodyJson)

    }, function(data) {

        cancelEdit();

        updateMain();

    });

}

function editRow() {

    if (activeRowIndex === null) {

        return;

    }

    const bodyJson = {

        job\_id: mainValues[activeRowIndex].job\_id,

        job\_name: document.querySelector('#job-name').value,

        salary: document.querySelector('#job-salary').value,

        address: document.querySelector('#job-address').value,

        email: document.querySelector('#job-email').value,

        skill\_id: document.querySelector('#select-skill').value,

        employmenttype\_id: document.querySelector('#select-type').value

    };

    userFetch(`api/jobs/upgrade.php`, {

        method: 'POST',

        body: JSON.stringify(bodyJson)

    }, function(data) {

        cancelEdit();

        updateMain();

    });

}

function cancelEdit() {

    setActiveRow(null);

    closeInputFields();

}

function removeRow(index) {

    const needToRemove = confirm('Вы точно хотите удалить данные?');

    if (!needToRemove) {

        return;

    }

    const id = mainValues[index].job\_id;

    if (needToRemove) {

        userFetch(`api/jobs/delete.php?job\_id=${id}`, null, function() {

            updateMain();

        });

        if (index == activeRowIndex) {

            closeInputFields();

        }

    }

}

function openInputFields() {

    let obj = document.getElementById("input-area-data");

    if (obj.style.display != 'flex') {

        obj.style.display = 'flex';

    }

}

function closeInputFields() {

    let obj = document.getElementById("input-area-data");

    if (obj.style.display != 'none') {

        obj.style.display = 'none';

    }

}

function showField() {

    document.querySelector('#input-area-title').innerHTML = 'Добавление новой записи';

    document.querySelector('#input-button').innerHTML = 'Добавить';

    document.querySelector('#input-button').setAttribute('onclick', 'addRow()');

    setActiveRow(null);

    openInputFields();

}

Upgrade.php

<?php

    include("../db\_connect.php");

    $entity\_body = file\_get\_contents('php://input');

    $post\_data = json\_decode($entity\_body, true);

    if (!$post\_data) {

        $result\_data = [

            "error" => 1,

            "errorMsg" => "Не переданы входные параметры."

        ];

        echo json\_encode($result\_data, JSON\_UNESCAPED\_UNICODE);

        die();

    }

    // update data

    if (array\_key\_exists('job\_id', $post\_data)) {

        $sql = "UPDATE jobs

                SET job\_name = :job\_name,

                       salary = :salary,

                       address = :address,

                       email = :email,

                       employmenttype\_id = :employmenttype\_id,

                       skill\_id = :skill\_id

                 WHERE job\_id = :job\_id;";

        $stmt = $pdo->prepare($sql);

        $stmt->bindParam(':job\_id', $post\_data['job\_id']);

    } else {

        $sql = "INSERT INTO jobs (job\_name, salary, address, email, employmenttype\_id, skill\_id)

                VALUES

                       (:job\_name, :salary, :address, :email, :employmenttype\_id, :skill\_id);";

        $stmt = $pdo->prepare($sql);

    }

    if ($post\_data['job\_name'] === "") {

        $result\_data = [

            "error" => 1,

            "errorMsg" => "Имя не может быть пустым."

        ];

        echo json\_encode($result\_data, JSON\_UNESCAPED\_UNICODE);

        die();

    }

    if ($post\_data['employmenttype\_id'] === "") {

        $result\_data = [

            "error" => 1,

            "errorMsg" => "Выберите тип занятости."

        ];

        echo json\_encode($result\_data, JSON\_UNESCAPED\_UNICODE);

        die();

    }

    if (!is\_numeric($post\_data['salary'])) {

        $result\_data = [

            "error" => 1,

            "errorMsg" => "Укажите число в поле зарплаты."

        ];

        echo json\_encode($result\_data, JSON\_UNESCAPED\_UNICODE);

        die();

    }

    elseif ($post\_data['salary'] > 1000000 || $post\_data['salary'] < 0){

        $result\_data = [

            "error" => 1,

            "errorMsg" => "Значение зарплаты некорректно"

        ];

        echo json\_encode($result\_data, JSON\_UNESCAPED\_UNICODE);

        die();

    }

    // Вычисляемое поле1

    if ($post\_data['skill\_id'] == 5 && $post\_data['employmenttype\_id'] == 1) {

        $post\_data['salary'] = 999999;

    }

    // Вычисляемое поле2

    if ($post\_data['salary'] == 0) {

        $post\_data['employmenttype\_id'] = 5;

    }

    elseif ($post\_data['salary'] < 15000){

        $post\_data['employmenttype\_id'] = 2;

    }

    if ($post\_data['salary'] === "") $post\_data['salary'] = null;

    if ($post\_data['address'] === "") $post\_data['address'] = null;

    if ($post\_data['email'] === "") $post\_data['email'] = null;

    if ($post\_data['skill\_id'] === "") $post\_data['skill\_id'] = null;

    $stmt->bindParam(':job\_name', $post\_data['job\_name']);

    $stmt->bindParam(':salary', $post\_data['salary']);

    $stmt->bindParam(':address', $post\_data['address']);

    $stmt->bindParam(':email', $post\_data['email']);

    $stmt->bindParam(':employmenttype\_id', $post\_data['employmenttype\_id']);

    $stmt->bindParam(':skill\_id', $post\_data['skill\_id']);

    echo get\_json($stmt);

?>

Read.php

<?php

    include("../db\_connect.php");

    $sql = "SELECT j.\*, s.skill\_name, et.employmenttype\_name

              FROM

                   jobs j

                   LEFT JOIN skills s USING(skill\_id)

                   LEFT JOIN employment\_types et USING(employmenttype\_id)

             ORDER BY j.job\_name;";

    $stmt = $pdo->prepare($sql);

    echo get\_json($stmt);

?>

Delete.php

<?php

    include("../db\_connect.php");

    $sql = "DELETE FROM jobs

             WHERE job\_id = :job\_id;";

    $stmt = $pdo->prepare($sql);

    $stmt->bindParam(':job\_id', $\_GET['job\_id']);

    echo get\_json($stmt);

?>

Gettypes.php

<?php

    include("../db\_connect.php");

    $sql = "SELECT employmenttype\_id, employmenttype\_name

              FROM employment\_types

             ORDER BY employmenttype\_name";

    $stmt = $pdo->prepare($sql);

    echo get\_json($stmt);

?>

Index.php

<?php

    session\_start();

    if (!array\_key\_exists('IS\_AUTH', $\_SESSION)) {

        header('Location: authorization.php');

        die();

    }

?>

<head>

    <link href="css/index.css" rel="stylesheet">

    <link href="css/header.css" rel="stylesheet">

    <title>Главная</title>

</head>

<body>

    <?php

        include('header.php');

    ?>

    <div class="main">

        <div class="table-container">

            <table border="1">

                <thead>

                    <tr>

                        <th>Имя</th>

                        <th>Зарплата</th>

                        <th>Должность</th>

                        <th>Почта</th>

                        <th>Рабочий отдел</th>

                        <th>Тип работы</th>

                    </tr>

                </thead>

                <tbody id="main-container">

                </tbody>

            </table>

        </div>

        <div class="input-area-and-button">

            <div class="input-area">

                <div id="input-area-data">

                    <h1 id="input-area-title"></h1>

                    <span>Имя: </span><input id="job-name"><br/>

                    <span>Зарплата: </span><input id="job-salary"><br/>

                    <span>Должность: </span><input id="job-address"><br/>

                    <span>Почта: </span><input type="email" id="job-email"><br/>

                    <span>Рабочий отдел: </span>

                    <select id="select-skill">

                        <option></option>

                    </select><br/>

                    <span>Тип работы: </span>

                    <select id="select-type">

                        <option></option>

                    </select><br/>

                    <div class="input-area-data-button">

                        <button id="input-button" onclick="addRow()">Добавить</button>

                        <button onclick="cancelEdit()">Отмена</button>

                    </div>

                </div>

            </div>

            <div class="button">

                <button class="open-input-button" onclick="showField()">Добавить</button>

            </div>

        </div>

    </div>

    <script src="js/script.js"></script>

</body>

Admin.php

<head>

    <title>Статистика</title>

    <link href="css/admin.css" rel="stylesheet">

    <link href="css/header.css" rel="stylesheet">

</head>

<?php

    session\_start();

    if (!array\_key\_exists('IS\_AUTH', $\_SESSION)) {

        header('Location: authorization.php');

        die();

    }

    $dsn = "pgsql:host=localhost;port=5432;dbname=lab6krok;";

    $pdo = new PDO($dsn, 'postgres', 'irjkf2211rkfcc');

?>

<?php

        include('header.php');

?>

<p>

Роли вашего пользователя:

</p>

<ul>

    <?php

        $sql = "SELECT r.name as name

                  FROM

                       roles r

                       JOIN user\_roles ur ON r.id = ur.role\_id

                       JOIN users u ON u.id = ur.user\_id

                 WHERE ur.user\_id = :user\_id;";

        $stmt = $pdo->prepare($sql);

        $stmt->bindParam(":user\_id", $\_SESSION["user\_id"]);

        $stmt->execute();

        $rows = $stmt->fetchAll();

        foreach ($rows as $row) {

            if ($row["name"] == 'Администратор')

                echo "<li><b>". $row["name"] ."</b></li>";

            else

                echo "<li>". $row["name"] ."</li>";

        }

    ?>

</ul>

<?php

    if (in\_array('ADM', $\_SESSION['roles']) || in\_array('STAT', $\_SESSION['roles'])) {

        $sql = "SELECT u.login, u.name, STRING\_AGG(r.name, ',') AS roles

            FROM roles r

            JOIN user\_roles ur ON r.id = ur.role\_id

            JOIN users u ON ur.user\_id = u.id

            GROUP BY u.login, u.name;";

        $stmt = $pdo->query($sql);

        $rows = $stmt->fetchAll();

        echo '<table border="1">';

        echo '<tr><td>Логин</td><td>Имя</td><td>Роль</td></tr>';

        foreach($rows as $row) {

            echo '<tr>';

            echo '<td>' . $row['login'] . '</td>';

            echo '<td>' . $row['name'] . '</td>';

            echo '<td>' . $row['roles'] . '</td>';

            echo '<tr>';

        }

            echo '</table>';

        $sql = "SELECT skill\_name, COUNT(\*) as cnt

                      FROM skills JOIN jobs USING(skill\_id)

                     GROUP BY skill\_name

                     ORDER BY 2 DESC;";

        $stmt = $pdo->query($sql);

        $rows = $stmt->fetchAll();

        echo '<table border="1">';

        echo '<tr><td>Навык</td><td>Кол-во</td></tr>';

        foreach($rows as $row) {

            echo '<tr>';

            echo '<td>' . $row['skill\_name'] . '</td>';

            echo '<td>' . $row['cnt'] . '</td>';

            echo '<tr>';

        }

            echo '</table>';

        $sql = "SELECT employmenttype\_name, COUNT(\*) as cnt

                      FROM employment\_types JOIN jobs USING(employmenttype\_id)

                     GROUP BY employmenttype\_name

                     ORDER BY 2 DESC;";

        $stmt = $pdo->query($sql);

        $rows = $stmt->fetchAll();

        echo '<table border="1">';

        echo '<tr><td>Тип</td><td>Кол-во</td></tr>';

        foreach($rows as $row) {

            echo '<tr>';

            echo '<td>' . $row['employmenttype\_name'] . '</td>';

            echo '<td>' . $row['cnt'] . '</td>';

            echo '<tr>';

        }

            echo '</table>';

        } else {

            echo '<div class="avail-warn"><h1>У вас недостаточно привилегий для

            просмотра данной страницы</h1></div>';

        }

?>

Authorization.php

<?php

    session\_start();

    if (array\_key\_exists("IS\_AUTH", $\_SESSION)) {

        header("Location: index.php");

        die();

    }

?>

<!DOCTYPE html>

<html lang="ru">

    <head>

        <title>Авторизация</title>

        <meta charset="utf-8"/>

        <link rel="stylesheet" href="css/style.css">

    </head>

    <body>

        <form class="form" action="/login.php" method="post">

            <h1 class="form\_title">Авторизация</h1>

            <div class="form\_group">

                <input class="form\_input" placeholder="Логин" name="login">

            </div>

            <div class="form\_group">

                <input class="form\_input" placeholder="Пароль" type="password" name="pass">

            </div>

            <div class="form\_button\_div">

                <input class="form\_button" type="submit" value="Ок">

            </div>

        </form>

    </body>

</html>

Find\_roles.php

<?php

    function get\_roles($user\_id) {

        $dsn = "pgsql:host=localhost;port=5432;dbname=lab6krok;";

        $pdo = new PDO($dsn, 'postgres', 'irjkf2211rkfcc');

        $sql = 'SELECT r.code

                  FROM user\_roles ur

                       JOIN roles r ON r.id = ur.role\_id

                 WHERE ur.user\_id = :user\_id;';

        $stmt = $pdo->prepare($sql);

        $stmt->bindParam(':user\_id', $user\_id);

        $stmt->execute();

        return $stmt->fetchAll(PDO::FETCH\_COLUMN, 0);

    }

?>

Login.php

<?php

    session\_start();

    if (array\_key\_exists('IS\_AUTH', $\_SESSION)) {

        header('Location: index.php');

        die();

    }

    $dsn = "pgsql:host=localhost;port=5432;dbname=lab6krok;";

    $pdo = new PDO($dsn, 'postgres', 'irjkf2211rkfcc');

    try {

        $stmt = $pdo->prepare("SELECT id, name, login FROM users WHERE login = :login

                               AND pass = :pass");

        $stmt->bindParam(':login', $\_POST['login']);

        $stmt->bindParam(':pass', $\_POST['pass']);

        $stmt->execute();

    } catch(Exception $e) {

        http\_response\_code(500);

        die();

    }

    $row = $stmt->fetch(PDO::FETCH\_ASSOC);

    if ($row) {

        include('find\_roles.php');

        $\_SESSION['roles'] = get\_roles($row['id']);

        $\_SESSION['IS\_AUTH'] = true;

        $\_SESSION['name'] = $row['name'];

        $\_SESSION['login'] = $row['login'];

        $\_SESSION['user\_id'] = $row['id'];

        header('Location: index.php');

        die();

    } else {

        readfile('html/login.html');

    }

?>

Logout.php

<?php

    session\_start();

    if (array\_key\_exists('IS\_AUTH', $\_SESSION)) {

        session\_destroy();

        header('Location: authorization.php');

        die();

    }

?>

Start.sh

#!/bin/bash

sudo service apache2 start

sudo service postgresql start

sudo pgadmin4&

stop.sh

#!/bin/sh

sudo service apache2 stop

sudo service postgresql stop

index.css

body,

h1,

h2,

h3,

h4,

p,

ul[class],

ol[class],

li,

figure,

figcaption,

blockquote,

dl,

dd {

  margin: 0;

}

.main {

    display: flex;

    flex-direction: row;

    width: 100%;

    justify-content: space-around;

    height: 80vh;

}

table, th, td {

    border-collapse: collapse;

    overflow-y: auto;

}

.table-container {

    height: 100%;

    border: 2px black solid;

    background: gainsboro;

    overflow: scroll;

}

.table-container  table {

    border: none;

    border-collapse: collapse;

}

.table-container thead  tr {

    height: 10px;

    background: none;

}

.table-container thead tr th {

    border: 2px solid #E4E4E4;

    height: 10px;

    padding: 3px 5px;

    font-size: 12px;

}

.table-container tbody {

    height: 100%;

}

.table-container tr {

    background-color: white;

    border-bottom: 5px solid #e6e6e6;

    height: 20px;

}

.table-container tbody td{

    border: none;

    height: 20px;

    padding: 3px 5px;

    font-size: 18px;

}

.activeRow {

    background: #FF6347!important;

}

#main-container tr td {

    cursor: pointer;

}

#main-container tr:hover {

    background: #FF6347;

}

.input-area-and-button {

    display: flex;

    flex-direction: column;

    justify-content: space-between;

    align-items: flex-start;

    height: 100%;

}

.button {

    margin: 0 auto;

    align-self: flex-end;

}

.input-area {

    height: 90%;

    margin: 0px 0px;

    padding: 0 70px;

    font-family: serif;

    letter-spacing: 1px;

    display: flex;

    justify-content: center;

    align-items: center;

    width: 300px;

    border: 2px rgb(5, 0, 0) solid;

    background: gainsboro;

}

#input-area-data {

    display: none;

    flex-direction: column;

    justify-content: space-between;

}

.input-area-title {

    text-align: center;

    font-weight: normal;

    padding-bottom: 20px;

}

Styles.css

body {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

.form {

    width: 300px;

    padding: 25px;

    border-radius: 20px;

    box-shadow: 0 4px 16px #ccc;

    font-family: serif;

    letter-spacing: 1px;

}

.form\_input,

.form\_button {

    font-family: serif;

    letter-spacing: 1px;

    font-size: 16px;

}

.form\_title {

    text-align: center;

    font-weight: normal;

    padding-bottom: 20px;

}

.form\_warn {

    color: red;

    font-size: 18px;

    text-align: center;

    margin-bottom: 30px;

}

.form\_group {

    position: relative;

    padding-bottom: 32px;

}

.form\_input {

    width: 100%;

    padding: 0 0 10px 0;

    border: none;

    border-bottom: 1px solid #e0e0e0;

    background-color: transparent;

    outline: none;

    transition: 0.3s;

}

.form\_input:focus {

    border-bottom: 1px solid #1a73a8;

}

.form\_button {

    width: 120px;

    padding: 10px 20px;

    display: flex;

    align-items: center;

    justify-content: center;

    border: none;

    border-radius: 5px;

    color: #fff;

    background-color: #FF6347;

    outline: none;

    cursor: pointer;

    transition: 0.3s;

}

.form\_button\_div {

    display: flex;

    justify-content: center;

}

.form\_button:focus,

.form\_button:hover {

    background-color: #FF8C00;

}

Admin.css

.avail-warn {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

.avail-warn h1{

    background-color: lightgray;

    width: 300px;

    height: 100px;

    color: red;

    align-self: flex-start;

    text-align: center;

    padding: 10px;

    border-radius: 5px;

    box-shadow: 0 4px 16px #ccc;

    font-family: serif;

    letter-spacing: 1px;

    font-family: serif;

    font-size: 20px;

}

table {

    border: 2px solid black;

    border-collapse: collapse;

    margin-bottom: 10px;

}

p {

    font-family: sans-serif;

    font-size: 20px;

}

ul {

    font-family: sans-serif;

    font-size: 15px;

}

Header.css

header {

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: center;

    background-color: #e0e0e0;

    margin-bottom: 30px;

    width: 100%;

}

.header-title {

    font-weight: bold;

    font-size: 30px;

    padding: 10px;

}

.header-logout {

    display: flex;

    flex-direction: row;

    justify-content: center;

    padding: 10px;

}

.header-logout-title {

    align-self: center;

}

.header-logout-button {

    width: 120px;

    padding: 10px 20px;

    margin-top: 18px;

    margin-left: 10px;

    border: none;

    border-radius: 5px;

    color: #fff;

    background-color: #FF6347;

    outline: none;

    cursor: pointer;

    transition: 1s;

}

.header-logout-button:focus,

.header-logout-button:hover {

    background-color: #FF8C00;

}

Db\_connect.php

<?php

    function get\_json($statement) {

        error\_reporting(0);

        session\_start();

        if (!array\_key\_exists('IS\_AUTH', $\_SESSION)) {

            $result\_data = [

                "error" => 1,

                "errorMsg" => "Вы не авторизованы."

            ];

            return json\_encode($result\_data, JSON\_UNESCAPED\_UNICODE);

        }

        header("Content-type: application/json");

        $result\_data = [];

        try {

            $statement->execute();

            $result\_list = $statement->fetchAll(PDO::FETCH\_ASSOC);

            $result\_data = [

                "error" => 0,

                "data" => $result\_list

            ];

        } catch (PDOException $e) {

            $result\_data = [

                "error" => 1,

                "errorMsg" => "Ошибка БД."

            ];

        }

        return json\_encode($result\_data, JSON\_UNESCAPED\_UNICODE);

    }

    $dsn = "pgsql:host=localhost;port=5432;dbname=lab6krok;";

    $pdo = new PDO($dsn, 'postgres', 'irjkf2211rkfcc');

?>