ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ст. преподаватель		Н.А. Соловьева
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

Асинхронное клиент-серверное взаимодействие с использованием Ajax и JSON

по курсу: Web-Технологии

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4132		Р.В.Шенин
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

1. Цель работы:

Изучение и применение на практике технологии AJAX

2. Задание

Вариант: 3

POST запрос

Модификация данных без перезагрузки страницы: при нажатии кнопки «сохранить», запись сохраняется в базе, но перезагрузка страницы не выполняется.

Базовая часть:

Разработать AJAX-приложение для доступа к базе данных, подготовленной в предыдущей лабораторной работе. Вариант задания и тип запроса указаны в таблице 1. Формулировку варианта адаптировать под свою базу данных. В сценарии javascript запрещается использование библиотек. Новая страница должна быть встроена в сайт, подготовленный в предыдущих разделах.

Расширенная часть:

1. использовать json

2. проверить работу в разных браузерах. В сценарии на javascript добавить вывод на страницу названия объекта, выполняющего асинхронный обмен, и название браузера. В отчете привести доказательства этой работы.

3. Адаптивная формулировка задания

Добавление данных в таблицу без перезагрузки страницы. При заполнении полей формы и нажатии кнопки добавить, запись сохраняется в базе данных, перезагрузка страницы не выполняется.

4. Структура таблиц БД

Таблица Jobs

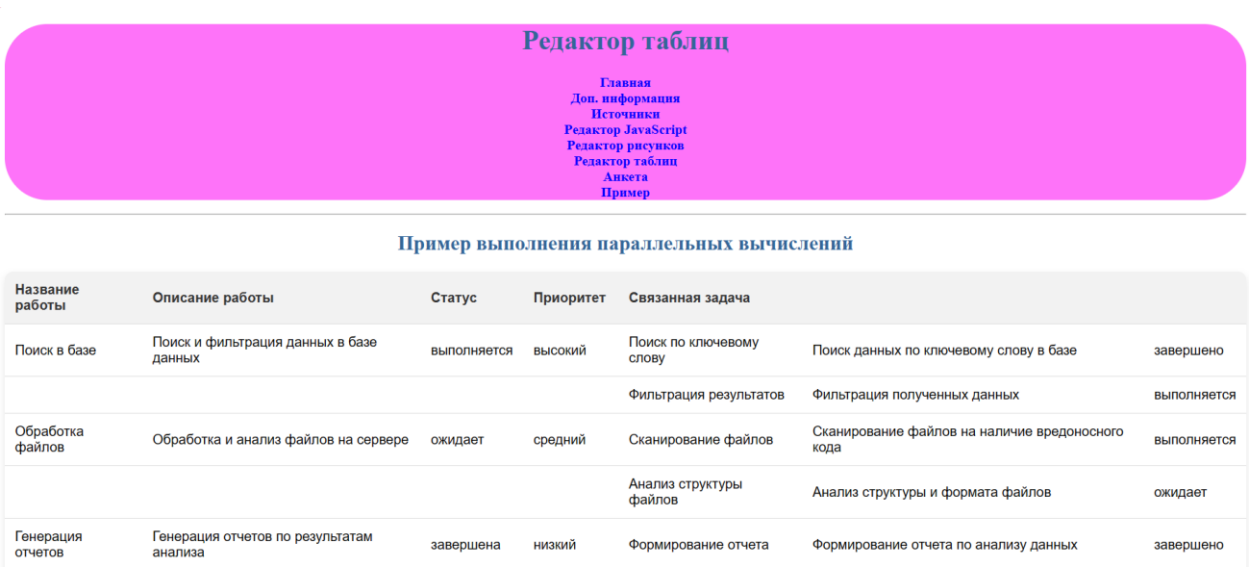
	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
☐	1	**job_id** 🔑	int(7)			Нет	*Нет*		AUTO_INCREMENT
☐	2	**job_name**	varchar(100)	utf8_general_ci		Нет	*Нет*		
☐	3	**job_description**	text	utf8_general_ci		Нет	*Нет*		
☐	4	**job_status**	varchar(100)	utf8_general_ci		Нет	*Нет*		
☐	5	**job_priority**	varchar(100)	utf8_general_ci		Нет	*Нет*		

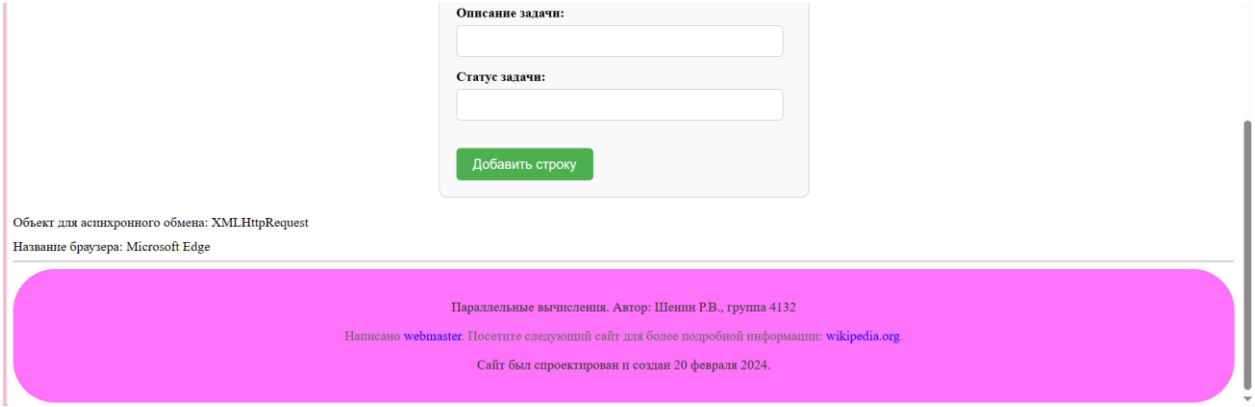
Таблица Tasks

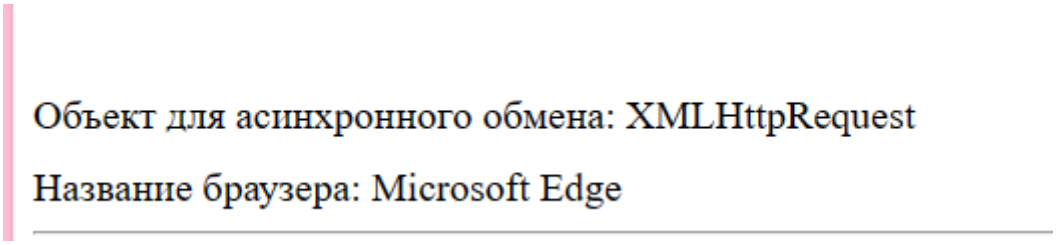
	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
☐	1	**task_id** 🔑	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить Удалить Ещё
☐	2	**job_id**	int(11)			Да	*NULL*			Изменить Удалить Ещё
☐	3	**task_name**	varchar(100)	utf8mb4_general_ci		Да	*NULL*			Изменить Удалить Ещё
☐	4	**task_description**	text	utf8mb4_general_ci		Да	*NULL*			Изменить Удалить Ещё
☐	5	**task_status**	varchar(100)	utf8mb4_general_ci		Да	*NULL*			Изменить Удалить Ещё

5. Скриншоты страниц сайта и таблиц базы данных

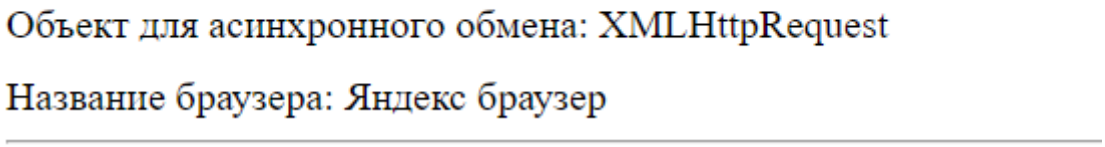
Сайт открыт в Edge







Сайт открыт в Яндекс



Скриншоты таблицы базы данных

Таблица Jobs

	job_id ▲ 1	job_name	job_description	job_status	job_priority
Изменить Копировать Удалить	1	Поиск в базе	Поиск и фильтрация данных в базе данных	выполняется	высокий
Изменить Копировать Удалить	2	Обработка файлов	Обработка и анализ файлов на сервере	ожидает	средний
Изменить Копировать Удалить	3	Генерация отчетов	Генерация отчетов по результатам анализа	завершена	низкий

Таблица Tasks

	task_id	job_id	task_name	task_description	task_status
Изменить Копировать Удалить	1	1	Поиск по ключевому слову	Поиск данных по ключевому слову в базе	завершено
Изменить Копировать Удалить	2	1	Фильтрация результатов	Фильтрация полученных данных	выполняется
Изменить Копировать Удалить	3	2	Сканирование файлов	Сканирование файлов на наличие вредоносного кода	выполняется
Изменить Копировать Удалить	4	2	Анализ структуры файлов	Анализ структуры и формата файлов	ожидает
Изменить Копировать Удалить	5	3	Формирование отчета	Формирование отчета по анализу данных	завершено

6. Листинг

Javascript код

```
<script type="text/javascript">
    var req = false;
    var ajaxObjectName = "";
    var browserName = "";
    //проверяем тип браузера и создаем объект
    if (window.XMLHttpRequest) {
       req = new XMLHttpRequest(); //Создаем объект
       ajaxObjectName = "XMLHttpRequest";
    } else {
       try {
          req = new ActiveXObject('Msxml2.XMLHTTP');
          ajaxObjectName = "Msxml2.XMLHTTP";
       } catch (e) {
          req = new ActiveXObject('Microsoft.XMLHTTP');
          ajaxObjectName = "Microsoft.XMLHTTP";
       }
    }




    if (!req) // если объект XMLHttpRequest не поддерживается
       alert('Объект XMLHttpRequest не поддерживается данным браузером');


       // Определение названия браузера
    if (navigator.userAgent.indexOf("Edg") !== -1) {
```

```
    browserName = "Microsoft Edge";
}
else {
    browserName = "Яндекс браузер";
}

// Отображение информации на странице
document.addEventListener("DOMContentLoaded", function() {
    document.getElementById("ajaxObjectName").innerText = "Объект для асинхронного обмена: " + ajaxObjectName;
    document.getElementById("browserName").innerText = "Название браузера: " + browserName;
});

    //после нажатия кнопки модификации
    function Load() {
        if (!req) return;

        //получаем данные с формы
        var jobName = document.getElementById("job_name").value;
        var jobDescription = document.getElementById("job_description").value;
        var jobStatus = document.getElementById("job_status").value;
        var jobPriority = document.getElementById("job_priority").value;
        var taskName = document.getElementById("task_name").value;
        var taskDescription = document.getElementById("task_description").value;
        var taskStatus = document.getElementById("task_status").value;

        //для отладки
```

```
        console.log("jobName:", jobName);
        console.log("jobDescription:", jobDescription);
        console.log("jobStatus:", jobStatus);
        console.log("jobPriority:", jobPriority);
        console.log("taskName:", taskName);
        console.log("taskDescription:", taskDescription);
        console.log("taskStatus:", taskStatus);

        //имя функции, которую запускаем после получения ответа от сервера
        req.onreadystatechange = receive;

        //создаем строку параметров для запроса методов POST
        var params = "job_name=" + encodeURIComponent(jobName) + //encode кодирует значения параметров, нужен для правильной для правильной передачи данных, содержащих специальные символы
                "&job_description=" + encodeURIComponent(jobDescription) +
                "&job_status=" + encodeURIComponent(jobStatus) +
                "&job_priority=" + encodeURIComponent(jobPriority) +
                "&task_name=" + encodeURIComponent(taskName) +
                "&task_description=" + encodeURIComponent(taskDescription) +
                "&task_status=" + encodeURIComponent(taskStatus);

        req.open("POST", "add_row_ajax.php", true); //открываем соединение
        req.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); //данные в теле POST-запроса будут закодированы в формате application/x-www-form-urlencoded.
        req.send(params); //передаем запрос серверу
    }
```

```
function receive() { //получение данных от сервера
    if (req.readyState == 4) { //если запрос завершен
        if (req.status == 200) { //если статус запрос - OK
            try {
                var response = JSON.parse(req.responseText); //разбор JSON строки от сервера
                if (response.error) {
                    alert(response.error);
                } else {
                    addRowToTable(response);
                }
            } catch (e) {
                console.error("Ошибка парсинга JSON:", e, req.responseText);
                alert("Ошибка на сервере. Проверьте консоль для деталей.");
            }
        } else {
            alert('Ошибка ' + req.status + ': \n' + req.statusText);
        }
    }
}

function addRowToTable(data) {
    var table = document.querySelector("table");
    var newRow = table.insertRow(-1); // Добавляем новую строку в конец таблицы

    var jobNameCell = newRow.insertCell(0);
    var jobDescriptionCell = newRow.insertCell(1);
    var jobStatusCell = newRow.insertCell(2);
```

```
        var jobPriorityCell = newRow.insertCell(3);
        var taskNameCell = newRow.insertCell(4);
        var taskDescriptionCell = newRow.insertCell(5);
        var taskStatusCell = newRow.insertCell(6);

        jobNameCell.textContent = data.job_name;
        jobDescriptionCell.textContent = data.job_description;
        jobStatusCell.textContent = data.job_status;
        jobPriorityCell.textContent = data.job_priority;
        taskNameCell.textContent = data.task_name;
        taskDescriptionCell.textContent = data.task_description;
        taskStatusCell.textContent = data.task_status;
        }
        </script>
```

PHP код

```
<?php

$link = mysqli_connect('localhost', 'root', '', 'parallelcomputing');

if (!$link) {
   printf("Невозможно подключиться к базе данных. Код ошибки: %s\n", mysqli_connect_error());
   exit;
}

$job_name = $_POST['job_name'] ?? '';
```

```
$job_description = $_POST['job_description'] ?? '';
$job_status = $_POST['job_status'] ?? '';
$job_priority = $_POST['job_priority'] ?? '';
$task_name = $_POST['task_name'] ?? '';
$task_description = $_POST['task_description'] ?? '';
$task_status = $_POST['task_status'] ?? '';

$sql = "INSERT INTO Jobs (job_name, job_description, job_status, job_priority)
        VALUES ('$job_name', '$job_description', '$job_status', '$job_priority')";

$response = array();

if (mysqli_query($link, $sql)) {  // Проверка успешного выполнения SQL запроса
    $job_id = mysqli_insert_id($link); // Получаем ID вставленной строки
    $sql_task = "INSERT INTO Tasks (job_id, task_name, task_description, task_status)
                VALUES ('$job_id', '$task_name', '$task_description', '$task_status')";
    if (mysqli_query($link, $sql_task)) {
        //создаем ассоциативный массив с данными, которые вернем клиенту
        $response = array(
            "job_name" => $job_name,
            "job_description" => $job_description,
            "job_status" => $job_status,
            "job_priority" => $job_priority,
            "task_name" => $task_name,
            "task_description" => $task_description,
            "task_status" => $task_status
        );
```

```
    } else {
        $response = array("error" => "Ошибка: " . mysqli_error($link));
    }
} else {
    $response = array("error" => "Ошибка: " . mysqli_error($link));
}

mysqli_close($link);

header('Content-Type: application/json'); //устанавливаем заголовок http ответа, указывающие что контент json
echo json_encode($response); //конвертируем массив в формат json
?>
```

HTML код

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 transitional//EN" "http://www.w3.org/TR/HTML4.01/loose.dtd">
<head>
    <title>Применение XMLHttpRequest</title>
    <meta charset="UTF8">

    <style>
        /* Стили для заголовков */
        h1, h2, h3 {
            color: #336699; /* Цвет текста */
        }

        /* Стили для списка ссылок в навигации */
```

```
nav ul li a {
    color: #333; /* Цвет текста */
    font-weight: bold; /* Жирный шрифт */
}

/* Стили для элемента заголовка */
.myclass1 {
    background-color: rgb(254, 115, 249); /* Цвет фона */
    border-radius: 50px; /* Радиус границ */
}

/* Стили для результата */
#result {
    width: 300px; /* Ширина */
}

/* Стили для ссылок */
a:link, a:visited {
    color: blue; /* Цвет ссылок */
    text-decoration: none; /* Отмена подчеркивания */
    transition: color 0.3s; /* Плавное изменение цвета */
}

/* При наведении на ссылку */
a:hover {
    color: #ff0000; /* Цвет ссылки при наведении */
}
```

```
/* Стили для списка */
nav ul {
    list-style: none; /* Убираем маркеры списка */
    padding: 0; /* Убираем отступы */
}

/* Стили для футера */
footer {
    text-align: center; /* Выравнивание текста по центру */
    padding: 20px 0; /* Внутренние отступы */
    background-color: rgb(254, 115, 249); /* Цвет фона */
    color: #333; /* Цвет текста */
    border-radius: 50px; /* Радиус границ */
}

/* Стили для адреса в футере */
footer address {
    font-style: normal; /* Отменяем курсив */
    color: #666; /* Цвет текста */
}

/* Стили для таблицы */
table {
    width: 100%; /* Ширина таблицы */
    border-collapse: collapse; /* Объединение границ */
    border-radius: 10px; /* Радиус границ */
```

```
    overflow: hidden; /* Скрытие содержимого, выходящего за границы */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Тень */
    font-family: Arial, sans-serif; /* Шрифт */
}

/* Стили для ячеек таблицы */
th, td {
    padding: 12px; /* Внутренние отступы */
    text-align: left; /* Выравнивание текста по левому краю */
    border-bottom: 1px solid #ddd; /* Граница снизу */
}

/* Стили для заголовков таблицы */
th {
    background-color: #f2f2f2; /* Цвет фона */
    color: #333; /* Цвет текста */
}

/* Стили для формы */
form {
    margin-top: 20px; /* Верхний отступ */
    width: 400px; /* Ширина */
    margin-left: auto; /* Автоматическое выравнивание по левому краю */
    margin-right: auto; /* Автоматическое выравнивание по правому краю */
    border: 1px solid #ccc; /* Граница */
    padding: 20px; /* Внутренние отступы */
    border-radius: 10px; /* Радиус границ */
```

```
    background-color: #f9f9f9; /* Цвет фона */
}

/* Стили для меток */
label {
    font-weight: bold; /* Жирный шрифт */
    margin-top: 10px; /* Верхний отступ */
}

/* Стили для полей ввода */
input[type="text"] {
    width: calc(100% - 10px); /* Ширина */
    padding: 10px; /* Внутренние отступы */
    margin-top: 5px; /* Верхний отступ */
    margin-bottom: 15px; /* Нижний отступ */
    border: 1px solid #ccc; /* Граница */
    border-radius: 5px; /* Радиус границ */
    box-sizing: border-box; /* Расчет размеров, включая границу и отступы */
}

/* Стили для кнопки отправки */
input[type="button"] {
    background-color: #4CAF50; /* Цвет фона */
    color: white; /* Цвет текста */
    padding: 10px 20px; /* Внутренние отступы */
    border: none; /* Граница */
    border-radius: 5px; /* Радиус границ */
```

```
        cursor: pointer; /* Изменение курсора при наведении */
        font-size: 16px; /* Размер шрифта */
    }

    /* При наведении на кнопку отправки */
    input[type="button"]:hover {
        background-color: #45a049; /* Цвет фона */
    }

</style>

<script type="text/javascript">
    var req = false;
    var ajaxObjectName = "";
    var browserName = "";
    //проверяем тип браузера и создаем объект
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest(); //Создаем объект
        ajaxObjectName = "XMLHttpRequest";
    } else {
        try {
            req = new ActiveXObject('Msxml2.XMLHTTP');
            ajaxObjectName = "Msxml2.XMLHTTP";
        } catch (e) {
            req = new ActiveXObject('Microsoft.XMLHTTP');
            ajaxObjectName = "Microsoft.XMLHTTP";
        }
```

```
}

if (!req) // если объект XMLHttpRequest не поддерживается
    alert('Объект XMLHttpRequest не поддерживается данным браузером');

    // Определение названия браузера
if (navigator.userAgent.indexOf("Edg") !== -1) {
    browserName = "Microsoft Edge";
}
else {
    browserName = "Яндекс браузер";
}

// Отображение информации на странице
document.addEventListener("DOMContentLoaded", function() {
    document.getElementById("ajaxObjectName").innerText = "Объект для асинхронного обмена: " + ajaxObjectName;
    document.getElementById("browserName").innerText = "Название браузера: " + browserName;
});

    //после нажатия кнопки модификации
    function Load() {
        if (!req) return;

        //получаем данные с формы
```

```
var jobName = document.getElementById("job_name").value;
var jobDescription = document.getElementById("job_description").value;
var jobStatus = document.getElementById("job_status").value;
var jobPriority = document.getElementById("job_priority").value;
var taskName = document.getElementById("task_name").value;
var taskDescription = document.getElementById("task_description").value;
var taskStatus = document.getElementById("task_status").value;

//для отладки
console.log("jobName:", jobName);
console.log("jobDescription:", jobDescription);
console.log("jobStatus:", jobStatus);
console.log("jobPriority:", jobPriority);
console.log("taskName:", taskName);
console.log("taskDescription:", taskDescription);
console.log("taskStatus:", taskStatus);

//имя функции, которую запускаем после получения ответа от сервера
req.onreadystatechange = receive;

//создаем строку параметров для запроса методов POST
var params = "job_name=" + encodeURIComponent(jobName) + //encode кодирует значения параметров, нужен для правильной для правильной передачи данных, содержащих специальные символы
    "&job_description=" + encodeURIComponent(jobDescription) +
    "&job_status=" + encodeURIComponent(jobStatus) +
    "&job_priority=" + encodeURIComponent(jobPriority) +
    "&task_name=" + encodeURIComponent(taskName) +
```

```
                "&task_description=" + encodeURIComponent(taskDescription) +
                "&task_status=" + encodeURIComponent(taskStatus);

            req.open("POST", "add_row_ajax.php", true); //открываем соединение
            req.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
//данные в теле POST-запроса будут закодированы в формате application/x-www-form-urlencoded.
            req.send(params); //передаем запрос серверу
        }

    function receive() { //получение данных от сервера
        if (req.readyState == 4) { //если запрос завершен
            if (req.status == 200) { //если статус запрос - OK
                try {
                    var response = JSON.parse(req.responseText); //разбор JSON строки от сервера
                    if (response.error) {
                        alert(response.error);
                    } else {
                        addRowToTable(response);
                    }
                } catch (e) {
                    console.error("Ошибка парсинга JSON:", e, req.responseText);
                    alert("Ошибка на сервере. Проверьте консоль для деталей.");
                }
            } else {
                alert('Ошибка ' + req.status + ': \n' + req.statusText);
            }
        }
```

```
        }

        function addRowToTable(data) {
            var table = document.querySelector("table");
            var newRow = table.insertRow(-1); // Добавляем новую строку в конец таблицы

            var jobNameCell = newRow.insertCell(0);
            var jobDescriptionCell = newRow.insertCell(1);
            var jobStatusCell = newRow.insertCell(2);
            var jobPriorityCell = newRow.insertCell(3);
            var taskNameCell = newRow.insertCell(4);
            var taskDescriptionCell = newRow.insertCell(5);
            var taskStatusCell = newRow.insertCell(6);

            jobNameCell.textContent = data.job_name;
            jobDescriptionCell.textContent = data.job_description;
            jobStatusCell.textContent = data.job_status;
            jobPriorityCell.textContent = data.job_priority;
            taskNameCell.textContent = data.task_name;
            taskDescriptionCell.textContent = data.task_description;
            taskStatusCell.textContent = data.task_status;
        }
    </script>
</head>
<body>
<body>
    <header class="myclass1" style="margin: 0 auto; text-align: center;">
```

```
    <h1>Редактор таблиц</h1>
    <nav>
        <ul>
            <li><a href="index.html">Главная</a></li>
            <li><a href="index2.html">Доп. информация</a></li>
            <li><a href="index3.html">Источники</a></li>
            <li><a href="third.html">Редактор JavaScript</a></li>
            <li><a href="third_dop.html">Редактор рисунков</a></li>
            <li><a href="index4.html">Редактор таблиц</a></li>
            <li><a href="forma.html">Анкета</a></li>
            <li><a href="tableBD.php">Пример</a></li>
        </ul>
    </nav>
</header>

<hr>

<h2 style="text-align: center;">Пример выполнения параллельных вычислений</h2>
<table id >
    <tr>
        <th>Название работы</th>
        <th>Описание работы</th>
        <th>Статус</th>
        <th>Приоритет</th>
        <th colspan="3">Связанная задача</th>
    </tr>
    <?php
```

```
/* Подключение к серверу MySQL */
$link = mysqli_connect(
   'localhost',  /* Хост, к которому мы подключаемся */
   'root',       /* Имя пользователя */
   '',   /* Используемый пароль */
   'parallelcomputing');     /* База данных для запросов по умолчанию */

if (!$link) {
   printf("Невозможно подключиться к базе данных. Код ошибки: %s\n", mysqli_connect_error());
   exit;
}

// Запрос на выборку данных из таблиц
$sql = "SELECT j.job_name, j.job_description, j.job_status, j.job_priority, t.task_name, t.task_description, t.task_status FROM Jobs j LEFT JOIN Tasks t ON j.job_id = t.job_id";
$result = $link->query($sql); //результат выполнения зарпоса

// Вывод данных в таблицу
if ($result->num_rows > 0) {
   // Используем ассоциативный массив для отслеживания, были ли уже выведены данные для работы
   $jobs = array();
   while($row = $result->fetch_assoc()) {
      // Если для этой работы еще не выведены данные, выводим их и помечаем, что они уже выведены
      if (!isset($jobs[$row["job_name"]])) {
         echo "<tr>";
```

```
                echo "<td>" . $row["job_name"]. "</td>";
                echo "<td>" . $row["job_description"]. "</td>";
                echo "<td>" . $row["job_status"]. "</td>";
                echo "<td>" . $row["job_priority"]. "</td>";
                $jobs[$row["job_name"]] = true; // Помечаем, что для этой работы уже выведены данные
            } else {
                // Иначе для этой работы уже выведены данные, пропускаем их
                printf("<td></td><td></td><td></td><td></td>");
            }
            // Выводим данные для задачи
            echo "<td>" . $row["task_name"]. "</td>";
            echo "<td>" . $row["task_description"]. "</td>";
            echo "<td>" . $row["task_status"]. "</td>";
            echo "</tr>";
        }
    } else {
        printf("<tr><td colspan='7'>%s</td></tr>". "Нет данных");
    }



    $link->close();
    ?>
</table>

<br><br>
```

```
<form action="add_row_ajax.php" method="post">
    <label for="job_name">Название работы:</label><br>
    <input type="text" id="job_name" name="job_name"><br>

    <label for="job_description">Описание работы:</label><br>
    <input type="text" id="job_description" name="job_description"><br>

    <label for="job_status">Статус:</label><br>
    <input type="text" id="job_status" name="job_status"><br>

    <label for="job_priority">Приоритет:</label><br>
    <input type="text" id="job_priority" name="job_priority"><br>

    <label for="task_name">Название задачи:</label><br>
    <input type="text" id="task_name" name="task_name"><br>

    <label for="task_description">Описание задачи:</label><br>
    <input type="text" id="task_description" name="task_description"><br>

    <label for="task_status">Статус задачи:</label><br>
    <input type="text" id="task_status" name="task_status"><br><br>

    <input type="button" value="Добавить строку" onclick="Load()"/>
</form>
```

```
    <div id="ajaxObjectName" style="margin-top: 20px;"></div>

    <div id="browserName" style="margin-top: 10px;"></div>


    <hr>


    <footer>

        <p>Параллельные вычисления. Автор: Шенин Р.В., группа 4132</p>

        <address>

            Написано <a href="mailto:webmaster@example.com">webmaster</a>.

            Посетите следующий сайт для более подробной информации: <a
href="https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB
%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D
0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F">wikipedia.org</a>.

        </address>

        <p>Сайт был спроектирован и создан <time datetime="2024-02-20T08:00">20 февраля
2024</time>.</p>

    </footer>

</body>

</body>

</html>
```