|  |  |
| --- | --- |
|  | МИНПРОСВЕЩЕНИЯ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  «Тульский государственный педагогический университет им. Л.Н. Толстого»  (ТГПУ им. Л.Н. Толстого) |

Институт передовых информационных технологий

ОТЧЕТ

ПО ИТОГАМ ПРОИЗВОДСТВЕННОЙ (ЭКСПЛУАТАЦИОННОЙ) ПРАКТИКИ

Выполнил:

студент 3 курса группы 1520321

Киселев Валерий Алексеевич

Руководитель практики:

доцент Филиппова Е. В.

Тула – 2025

Содержание

[Цель и задачи учебной практики 3](#_Toc196395746)

[Описание видов и направлений деятельности организации или подразделения 4](#_Toc196395747)

[Структура ООО «Максимастер» 5](#_Toc196395748)

[Аппаратное и программное обеспечение, используемое в работе подразделения или организации 6](#_Toc196395749)

[Основные задачи, решаемые в течение учебной практики 7](#_Toc196395750)

[Краткие выводы студента по прохождению учебной практики 39](#_Toc196395751)

[Сведения о ходе выполнения заданий студентом в процессе прохождения практики 40](#_Toc196395752)

# Цель и задачи учебной практики

Цель учебной (эксплуатационной) практики – подготовка к решению производственных задач, возникающих в процессе работы базы практики; закрепление и углубление теоретических знаний, полученных при изучении дисциплин учебного плана; приобретение и развитие необходимых практических умений и навыков в соответствии с требованиями к уровню подготовки выпускника; повышение уровня освоения компетенций в профессиональной деятельности.

**Задачами практики** являются:

1. *Ознакомление*:

• с организацией деятельности подразделения или организации-базы практики;

• с основными задачами в ИТ-сфере, решаемыми в рамках организации или подразделения;

• с используемыми в процессе работы подразделения или организации информационными технологиями и программным обеспечением.

1. *Приобретение практических навыков*:

• выполнения функциональных обязанностей;

• разработки документации;

• настройки и тестирование программного обеспечения;

• технического сопровождения ПО;

• практической реализации предлагаемых программных и технологических решений.

1. *Подготовка отчетной документации по итогам практики.*

# Описание видов и направлений деятельности организации или подразделения

Адрес организации: 300041, Тульская область, город Тула, Советская ул., д. 31

ИНН:7106071124

**Основные направления организации:**

* Разработка крупных интернет-магазинов
* Внедрение порталов и CRM на Битрикс 24
* Разработка B2B систем
* Разработка порталов и сервисов
* Техподдержка и доработка веб-проектов
* Технический аудит сайтов
* Автоматизация тестирования
* Администрирование веб-серверов
* Настройка веб-серверов
* Мониторинг веб-проектов
* Внешний бэкап-сервер
* Хостинг

**Основные направления деятельности по ОКВЭД:**

62.01 Разработка компьютерного программного обеспечения

**Дополнительные виды деятельности**

62.02 Деятельность консультативная и работы в области компьютерных технологий

62.09 Деятельность, связанная с использованием вычислительной техники и информационных технологий, прочая

63.11 Деятельность по обработке данных, предоставление услуг по размещению информации и связанная с этим деятельность

63.11.1 Деятельность по созданию и использованию баз данных и информационных ресурсов

# Структура ООО «Максимастер»

Предметом основной деятельности учреждения является разработка технически сложных и высоконагруженных веб-проектов

Отделы организации:

* HR-отдел
* Тестирования
* Веб-разработки
* Поддержки

# Аппаратное и программное обеспечение, используемое в работе подразделения или организации

Аппаратное обеспечение:

1. Рабочие станции разработчиков:

* Intel Core i5 12400f
* 32 ГБ ОЗУ
* SSD 1024 ГБ
* NVIDIA GeForce RTX 3050

1. Серверная инфраструктура:

* Серверы на базе Linux для развертывания приложений

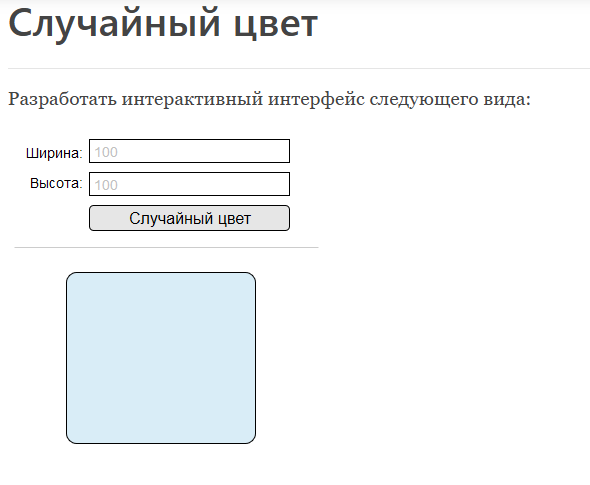
**Программное обеспечение и инструменты**:

1. PHP – язык разработки серверной части веб - приложений
2. PostgreSQL – свободная объектно-реляционная система управления базами данных
3. GitHub – облачная платформа для хостинга IT-проектов и совместной разработки. Основана на системе контроля версий Git
4. Redmine - открытое серверное веб-приложение для управления проектами и задачами
5. Postman – приложение для работы с API, которое позволяет создавать, тестировать и документировать API с помощью конструкторов запросов, визуализации ответов и автоматизации проверки
6. JavaScript- язык программирования, который остаётся основным инструментом для создания интерактивных и динамичных веб-приложений
7. Html – язык гипертекстовой разметки
8. Сss -  формальный язык декорирования и описания внешнего вида документа, написанного с использованием языка гипертекстовой разметки.

# Основные задачи, решаемые в течение учебной практики

1. Освоение инструмента для контроля версий Git и платформы GitHub
2. Освоение библиотеки Chart.js для построения графиков и диаграмм
3. Освоение хранилища данных LocalStorage
4. Освоение Composer для написания собственного API
5. Выполнение 9 практических задач

**Задача 1:**



**Решение:**

**CSS:**

#color-box {

  width: 150px;

  height: 150px;

  border: 1px solid black;

  margin-bottom: 50px;

  border-radius: 10px;

  margin-left: 80px;

  margin-top: 20px;

}

**Html:**

<!DOCTYPE html>

<html lang="ru">

<br><br><br>

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Случайный цвет</title>

  <link rel="stylesheet" href="RandomColor .css">

</head>

<body>

<div class="form-group" style="display: flex; flex-direction: column;">

    <div style="display: flex; align-items: center; margin-bottom: 5px;">

      <label for="Weight" style="width: 80px; text-align: right; margin-right: 5px; flex-shrink: 0;">Ширина:</label>

      <input type="text" name="Weight" id="Weight" style="max-width: 200px; box-sizing: border-box;">

    </div>

    <div style="display: flex; align-items: center; margin-bottom: 5px;">

      <label for="Height" style="width: 80px; text-align: right; margin-right: 5px; flex-shrink: 0;">Высота:</label>

      <input type="text" name="Height" id="Height" style="max-width: 200px; box-sizing: border-box;">

    </div>

    <input type="button" name="randomcolor" value="Случайный цвет" id="randomcolor" style="max-width: 178px; margin-left: 85px; ">

    </script>

  </div>

  <div id="color-box"></div>

  </div>

  <script src="RandomColor.js">

  </script>

</body>

</html>

**JS:**

const randomcolorButton = document.getElementById("randomcolor");

  randomcolorButton.onclick = function(event) {

    changeColor();

  };

  function getRandomColor() {

    const letters = '0123456789ABCDEF';

    let color = '#';

    for (let i = 0; i < 6; i++) {

      color += letters[Math.floor(Math.random() \* 16)];

    }

    return color;

  }

  function changeColor() {

    const colorBox = document.getElementById('color-box');

    colorBox.style.backgroundColor = getRandomColor();

    let width = document.getElementById("Weight").value;

    let height = document.getElementById("Height").value;

    if (!isNaN(width) && width > 0) {

      colorBox.style.width = width + "px";

    } else {

      alert("Некорректный ввод ширины");

      document.getElementById("Weight").value = "100";

    }

    if (!isNaN(height) && height > 0) {

      colorBox.style.height = height + "px";

    } else {

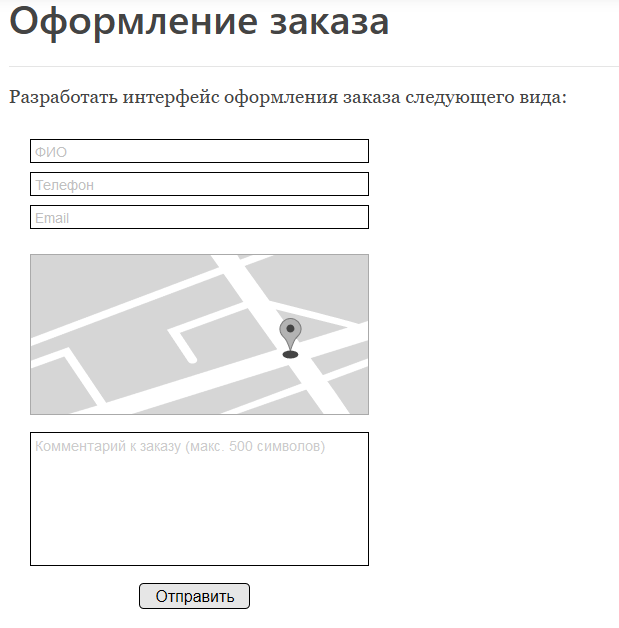
      alert("Некорректный ввод высоты");

      document.getElementById("Height").value = "100";

    }

  }

**Задача 2:**

****

**Решение:**

**CSS:**

       label {

            display: block;

            margin-bottom: 5px;

        }

        input[type="text"], textarea {

            width: 400px;

            padding: 8px;

            margin-bottom: 10px;

            border: 1px solid black;

            box-sizing: border-box;

        }

        textarea {

            height: 200px;

            resize: vertical;

        }

        #map {

            width: 400px;

            height: 200px;

            border: 1px solid black;

            margin-bottom: 10px;

        }

        input[type="submit"] {

            background-color: #eee;

            color: black;

            padding: 10px 20px;

            border: 1px solid black;

            cursor: pointer;

            text-align: center;

        }

        input[type="submit"]:hover {

            background-color: #ddd;

        }

        #orderForm {

            margin: 20px;

        }

**Html:**

<!DOCTYPE html>

<html lang="ru">

  <head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Оформление заказа</title>

    <link rel="stylesheet" href="Order.css">

    <script src="https://api-maps.yandex.ru/2.1/?apikey=913c846b-8db3-4642-b127-3a771adbcd28&lang=ru\_RU" type="text/javascript">

    </script>

    <script type="text/javascript" src="ymaps.js">

      </script>

    </head>

    <body>

      <form id="orderForm">

        <label for="fio"> </label>

        <input type="text" placeholder="ФИО" id="fio">

        <label for="phone"> </label>

        <input type="text" placeholder="Телефон" id="phone">

        <label for="email"> </label>

        <input type="text" placeholder="Email" id="email">

        <div id="map" ></div>

        <label for="comment"> </label>

        <textarea placeholder="Комментарий к заказу (макс. 500 символов)" id="comment"></textarea>

</form>

<input type="submit" id="submit" value="Отправить" style="margin-left: 150px;">

  <script src="Order.js">

    </script>

</body>

</html>

**JS:**

const submitbutton = document.getElementById("submit");

submitbutton.onclick = function(event){

  if (!checkData()) {

    event.preventDefault();

  } else {

    alert ("Форма отправлена");

  }

};

function checkData() {

      let fioInput = document.getElementById("fio");

      let fio = fioInput.value;

      let phoneInput = document.getElementById("phone");

      let phone = phoneInput.value;

      let emailInput = document.getElementById ("email");

      let email = emailInput.value;

      let commentInput = document.getElementById("comment");

      let comment = commentInput.value;

      if (fio.trim() === "") {

          alert("ФИО - обязательное поле");

          fioInput.focus();

          return false;

      }

      if (phone.trim() === "") {

          alert("Телефон - обязательное поле");

          phoneInput.focus();

          return false;

      }

      const lettersOnly = /^[а-яА-ЯёЁ\s]+$/;

      const numbersOnly = /^[0-9]+$/;

      if (!lettersOnly.test(fio)) {

          alert("ФИО должно содержать только буквы и пробелы");

          fioInput.value = "";

          fioInput.focus();

          return false;

      }

      if (!numbersOnly.test(phone)) {

          alert("Номер телефона должен содержать только числа");

          phoneInput.value = "";

          phoneInput.focus();

          return false;

      }

      if (email.indexOf("@") === -1) {

      alert("Email должен содержать символ @");

      emailInput.focus();

      return false;

      }

      if (comment.length > 500) {

      alert("Ваш комментарий более 500 символов");

      commentInput.focus();

      return false;

    }

      return true;

  }

**Ymap.js**

ymaps.ready(init);

      let myMap;

      let placemark;

      function init(){

          var myMap = new ymaps.Map("map", {

              center: [54.193122, 37.617348],

              zoom: 12

          });

          myMap.events.add('click', function (e) {

              const coords = e.get('coords');

              if (placemark) {

                  placemark.geometry.setCoordinates(coords);

              } else {

                  placemark = new ymaps.Placemark(coords);

                  myMap.geoObjects.add(placemark);

              }

              placemark.properties.set('balloonContent', `Координаты: ${coords[0].toFixed(6)}, ${coords[1].toFixed(6)}`);

              myMap.balloon.open(coords, {

                  content: `Координаты: ${coords[0].toFixed(6)}, ${coords[1].toFixed(6)}`

              });

          });

      }

**Задача 3:**

****

**Решение:**

**CSS:**

body { font-family: sans-serif; }

#price-filter {

    margin-bottom: 10px;

    text-align: center;

    margin-right: 100px;

}

#price-filter input, #price-filter button {

    margin-left: 5px;

}

table {

    border-collapse: collapse;

    width: 40%;

    margin: 0 auto;

}

th, td {

    border: 1px solid #ddd;

    padding: 8px;

    text-align: center;

}

th {

    background-color: #f2f2f2;

}

#message {

    margin-top: 10px;

    font-style: italic;

    color: #888;

    text-align: center;

}

.loader {

    text-align: center;

    margin-top: 20px;

}

.error-input {

    border: 1px solid red;

}

**Html:**

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Таблица товаров</title>

    <link rel="stylesheet" href="Tableofproducts.css">

</head>

<body>

<div id="price-filter">

    Цена от: <input type="number" id="price-from" value="0">

    до: <input type="number" id="price-to" value="10000">

    <button id="apply-filter">Обновить</button>

</div>

<table id="product-table">

    <thead>

        <tr>

            <th>ID</th>

            <th>Название</th>

            <th>Количество</th>

            <th>Цена за единицу</th>

            <th>Сумма</th>

        </tr>

    </thead>

    <tbody>

    </tbody>

</table>

<div id="message"></div>

<div class="loader" id="loader">Загрузка данных...</div>

<script src="Tableofproducts.js">

</script>

</body>

</html>

**JS:**

const API\_URL = "http://exercise.develop.maximaster.ru/service/products";

const proxyUrl = 'https://cors-anywhere.herokuapp.com/';

let table = document.getElementById("product-table");

let tbody = table.querySelector("tbody");

let priceFromInput = document.getElementById("price-from");

let priceToInput = document.getElementById("price-to");

let applyFilterButton = document.getElementById("apply-filter");

let messageDiv = document.getElementById("message");

let loaderDiv = document.getElementById("loader");

const username = 'cli';

const password = '12344321';

let productsData = [];

function toggleLoader(show) {

    loaderDiv.style.display = show ? "block" : "none";

}

function showMessage(text) {

    messageDiv.textContent = text;

    messageDiv.style.display = "block";

    table.style.display = "none";

}

function clearMessage() {

    messageDiv.textContent = "";

    messageDiv.style.display = "none";

    table.style.display = "table";

}

function applyFilter() {

clearMessage();

const priceFromValue = priceFromInput.value;

const priceToValue = priceToInput.value;

if (isNaN(priceFromValue) || isNaN(priceToValue)) {

    showMessage("Ошибка: Введите корректные числовые значения.");

    return;

}

const priceFrom = parseFloat(priceFromValue);

const priceTo = parseFloat(priceToValue);

if (priceFrom === 0 && priceTo === 0) {

    displayProducts(productsData);

    return;

}

if (!isNaN(priceFrom) && !isNaN(priceTo) && priceFrom > priceTo) {

    showMessage("Ошибка: Цена 'от' не может быть больше цены 'до'.");

    return;

}

const filteredProducts = productsData.filter(product => {

    const price = product.price;

    return price >= priceFrom && price <= priceTo;

});

displayProducts(filteredProducts);

}

function displayProducts(products) {

    tbody.innerHTML = "";

    if (products.length === 0) {

        showMessage("Нет данных, попадающих под условие фильтра");

        return;

    }

    products.forEach((product, index) => {

        const row = document.createElement("tr");

        const idCell = document.createElement("td");

        idCell.textContent = index + 1;

        const nameCell = document.createElement("td");

        nameCell.textContent = product.name;

        const quantityCell = document.createElement("td");

        quantityCell.textContent = product.quantity;

        const priceCell = document.createElement("td");

        priceCell.textContent = product.price;

        const sumCell = document.createElement("td");

        const sum = product.quantity \* product.price;

        sumCell.textContent = sum.toFixed(2);

        row.appendChild(idCell);

        row.appendChild(nameCell);

        row.appendChild(quantityCell);

        row.appendChild(priceCell);

        row.appendChild(sumCell);

        tbody.appendChild(row);

    });

    table.style.display = "table";

}

async function fetchproducts() {

    toggleLoader(true);

    try {

        const encodedCredentials = btoa(`${username}:${password}`);

        const response = await fetch(proxyUrl + API\_URL, {

            method: 'GET',

            headers: {

                'Authorization': `Basic ${encodedCredentials}`

            }

        });

        if (!response.ok) {

            throw new Error(`Ошибка HTTP: ${response.status}`);

        }

        productsData = await response.json();

        applyFilter();

    } catch (error) {

        console.error("Ошибка при получении данных:", error);

        showMessage(`Ошибка при получении данных: ${error.message}`);

    } finally {

        toggleLoader(false);

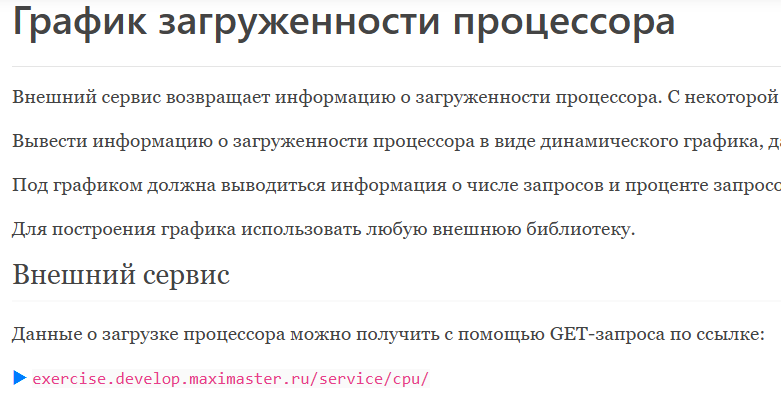
    }

}

applyFilterButton.addEventListener("click", applyFilter);

fetchproducts();

**Задача 4:**

****

**Решение:**

**CSS:**

#loader { display: none; text-align: center; margin-top: 10px; }

#request-stats { margin-top: 10px; }

**Html:**

<!DOCTYPE html>

<html>

<head>

    <title>График загруженности процессора</title>

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <link rel="stylesheet" href="LoadingProc.css">

</head>

<body>

<canvas id="myChart" width="400" height="200"></canvas>

<div id="loader">Загрузка...</div>

<div id="request-stats">

    Всего запросов: <span id="total-requests"> 0</span><br>

    Ошибочных запросов: <span id="error-requests">0</span> (<span id="error-percentage">0%</span>)

</div>

<script src="Loadingproc.js">

</script>

</body>

</html>

**JS:**

let cpuData = [];

let Numberofwrongzaprosov = 0;

let Numberofzaprosov = 0;

let Percentageofwrongzaprosov = 0;

let LastNotNullValue = 0;

const username = 'cli';

const password = '12344321';

const encodedCredentials = btoa(`${username}:${password}`);

const proxyUrl = 'https://cors-anywhere.herokuapp.com/';

const apiUrl = 'http://exercise.develop.maximaster.ru/service/cpu';

const totalRequestsElement = document.getElementById("total-requests");

const errorRequestsElement = document.getElementById ("error-requests");

const errorPercentageElement = document.getElementById ("error-percentage");

const ctx = document.getElementById('myChart').getContext('2d');

const MAX\_DATA\_POINTS = 20;

let myChart = new Chart(ctx, {

    type: 'line',

    data: {

        labels: [],

        datasets: [{

            label: 'Загруженность процессора в (%)',

            data: cpuData,

            borderColor: 'rgb(138, 43, 226)',

            tension: 0.1,

            fill: false

        }]

    },

    options: {

        responsive: true,

        scales: {

            y: {

                beginAtZero: true,

                max: 100

            }

        }

    }

});

function updateChart() {

    const labels = cpuData.map((\_, i) => `${i \* 5}s`);

    myChart.data.labels = labels;

    myChart.data.datasets[0].data = cpuData;

    myChart.update();

}

async function fetchCpuLoadProc() {

    try {

        const response = await fetch(proxyUrl + apiUrl, {

            method: 'GET',

            headers: {

                'Authorization': `Basic ${encodedCredentials}`

            }

        });

        if (!response.ok) {

            throw new Error(`Ошибка HTTP: ${response.status}`);

        }

        const data = await response.json();

        const cpuLoad = Number(data);

        if (cpuLoad === 0 ){

        Numberofwrongzaprosov +=1;

        cpuLoad = LastNotNullValue;

        } else {

            LastNotNullValue = cpuLoad;

        }

        cpuData.push(cpuLoad);

        Numberofzaprosov += 1;

        totalRequestsElement.textContent = Numberofzaprosov;

        errorRequestsElement.textContent = Numberofwrongzaprosov;

        Percentageofwrongzaprosov = (Numberofwrongzaprosov \*100)/Numberofzaprosov;

        errorPercentageElement.textContent = Percentageofwrongzaprosov;

        if (cpuData.length > MAX\_DATA\_POINTS) {

            cpuData.shift();

            console.warn("Была получена ошибка(0),используем предыдущее значение:", LastNotNullValue);

        }

        console.log("Массив данных:", cpuData);

        updateChart();

    } catch (error) {

        console.error('Ошибка при получении данных:', error);

    }

}

document.addEventListener('DOMContentLoaded', () => {

    fetchCpuLoadProc();

    setInterval(fetchCpuLoadProc, 5000);

});

**Задача 5:**

****

**Решение:**

**CSS:**

table {

    border-collapse: collapse;

    margin-right: 40px;

  }

  td {

    border: 1px solid black;

    padding: 15px;

    min-width: 150px;

    text-align: center;

  }

  .edit-mode {

    padding: 0;

  }

  .edit-mode input {

    width: 100%;

    height: 100%;

    box-sizing: border-box;

    border: none;

    text-align: center;

  }

  .button-column {

    display: flex;

    flex-direction: column;

    margin-right: 10px;

    margin-top: 50px;

  }

  .button-column button{

    margin: 3px 0px;

  }

  .Row-buttons {

    display: flex;

    justify-content: left;

    margin-top: 10px;

    margin-left: 200px;

  }

  .Row-buttons button {

    margin: 0px 3px;

  }

**Html:**

<!DOCTYPE html>

<html>

<head>

  <title>Электронная таблица</title>

  <link rel="stylesheet" href="ElectronicTable.css">

  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

</head>

<body>

  <div style="display: flex;">

    <table id="myTable">

      <tr>

        <td></td>

        <td></td>

        <td></td>

      </tr>

      <tr>

        <td></td>

        <td></td>

        <td></td>

      </tr>

    </table>

    <div class="button-column">

      <button onclick="addColumntable()">+</button>

      <button onclick="removeColumntable()">-</button>

    </div>

  </div>

  <div class="Row-buttons">

    <button onclick="addRowtable()">+</button>

    <button onclick="removeRowtable()">-</button>

  </div>

  <script src="ElectronicTable.js">

  </script>

</body>

</html>

**JS:**

let table = $('#myTable');

let localStorageKey = 'myTableData';

function loadDatalocalStorage() {

  let data = localStorage.getItem(localStorageKey);

  if (data) {

    table.html(data);

  }

}

function saveDatalocalStorage() {

  localStorage.setItem(localStorageKey, table.html());

}

function enableEdit(td) {

  let text = td.text();

  td.addClass('edit-mode').html('<input type="text" value="' + text + '">');

  let input = td.find('input').focus();

  input.on('blur keydown', function(e) {

    if (e.type === 'blur' || e.key === 'Enter') {

      let newText = input.val();

      td.removeClass('edit-mode').text(newText);

      saveData();

    }

  });

}

table.on('dblclick', 'td', function() {

  enableEdit($(this));

});

function addRowtable() {

  let numCols = table.find('tr:first-child td').length;

  let row = $('<tr></tr>');

  for (let i = 0; i < numCols; i++) {

    row.append('<td></td>');

  }

  table.append(row);

  saveData();

}

function removeRowtable() {

  if (table.find('tr').length > 1) {

    let lastRow = table.find('tr:last-child');

    let hasData = false;

    lastRow.find('td').each(function() {

      if ($(this).text().trim() !== '') {

        hasData = true;

        return false;

      }

    });

    if (hasData && !confirm("Удалить строку с данными?")) {

      return;

    }

    lastRow.remove();

    saveData();

  } else {

    alert("Вы не можете удалить последнюю строку.");

  }

}

function addColumntable() {

  table.find('tr').each(function() {

    $(this).append('<td></td>');

  });

  saveData();

}

function removeColumntable() {

  let numCols = table.find('tr:first-child td').length;

  if (numCols > 1) {

    let hasData = false;

    table.find('tr').each(function() {

      let lastCell = $(this).find('td:last-child');

      if (lastCell.text().trim() !== '') {

        hasData = true;

        return false;

      }

    });

    if (hasData && !confirm("Удалить столбец с данными?")) {

      return;

    }

    table.find('tr').each(function() {

      $(this).find('td:last-child').remove();

    });

    saveData();

  } else {

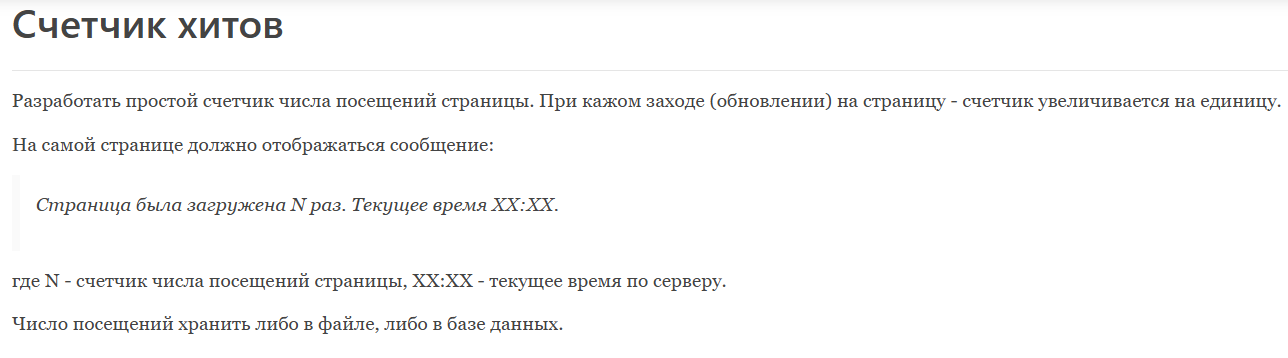
    alert("Вы не можете удалить последний столбец.");

  }

}

loadData();

**Задача 6:**

****

**Решение:**

**PHP + Html**

<!DOCTYPE html>  
<head>  
 <title>Счетчик хитов</title>  
</head>  
<body>  
  
  
<?php  
date\_default\_timezone\_set('Europe/Moscow');  
$counter\_file = 'Countofpage.txt';  
  
if (!file\_exists($counter\_file)) {  
 file\_put\_contents($counter\_file, '0');  
}  
  
$count = file\_get\_contents($counter\_file);  
$count++;  
  
file\_put\_contents($counter\_file, $count);  
?>  
  
<p><b>Страница была загружена <?php echo $count; ?> раз. Текущее время: <?php echo date('H:i:s'); ?></p>  
</body>

**Задача 7:**

****

**Решение:**

**CSS:**

body {

            font-family: Arial, sans-serif;

            max-width: 800px;

            margin: 0 auto;

            padding: 20px;

            line-height: 1.6;

            color: #333;

            background-color: #f9f9f9;

        }

        .message {

            border: 1px solid #ccc;

            padding: 15px;

            margin-bottom: 15px;

            background-color: #fff;

        }

        .message-date {

            color: #666;

            font-size: 0.9em;

            margin-bottom: 5px;

        }

        .message-author {

            font-weight: bold;

            margin-bottom: 5px;

        }

        form {

            margin-top: 30px;

            padding: 20px;

            background: #fff;

            border: 1px solid #ccc;

        }

        label {

            display: block;

            margin-bottom: 5px;

            font-weight: bold;

        }

        input[type="text"]

        {

            width: 50%;

            padding: 8px;

            margin-bottom: 15px;

            border: 1px solid #ccc;

            box-sizing: border-box;

        }

        textarea {

            width: 100%;

            padding: 8px;

            margin-bottom: 15px;

            border: 1px solid #ccc;

            box-sizing: border-box;

            height: 100px;

            resize: vertical;

        }

        button {

            background: #666;

            color: white;

            padding: 10px 15px;

            border: none;

            cursor: pointer;

            margin: 0 auto;

            display: block;

        }

        button:hover {

            background: #555;

        }

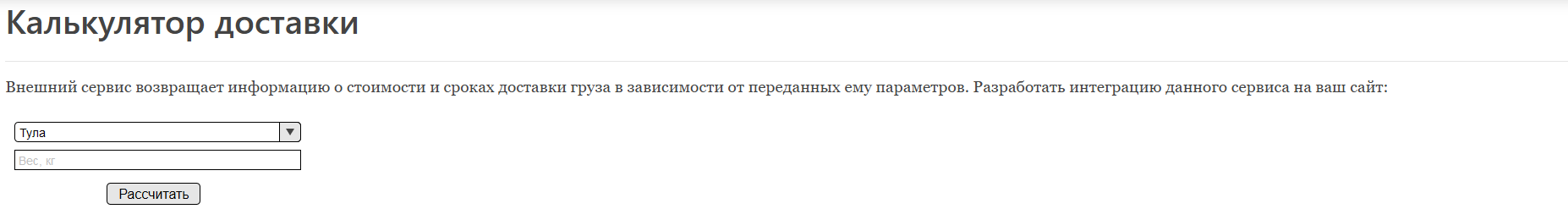
**Bookofguests.php**

<?php  
date\_default\_timezone\_set('Europe/Moscow');  
try {  
 $db = new PDO('sqlite:Bookofguests.db');  
 $db->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  
} catch (PDOException $e) {  
 die("Не удалось подключиться к базе данных: " . $e->getMessage());  
}  
  
  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  
 $name = trim($\_POST['name']);  
 $message = trim($\_POST['message']);  
  
 if (!empty($message)) {  
 if (empty($name)) {  
 $name = "Анонимно";  
 }  
  
 $stmt = $db->prepare('INSERT INTO Bookofguests (name, message) VALUES (:name, :message)');  
 $stmt->bindValue(':name', $name, PDO::PARAM\_STR);  
 $stmt->bindValue(':message', $message, PDO::PARAM\_STR);  
 $stmt->execute();  
 }  
}  
  
$results = $db->query('SELECT \* FROM Bookofguests ORDER BY created\_at DESC');  
$messages = $results->fetchAll(PDO::*FETCH\_ASSOC*);  
?>  
<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Гостевая книга</title>  
 <link rel="stylesheet" href="Bookofguests.css">  
</head>  
<body>  
<?php foreach ($messages as $msg): ?>  
 <div class="message">  
 <div class="message-date"><?php echo date('d.m.Y H:i', strtotime($msg['created\_at']) + 3 \* 60 \* 60); ?></div>  
 <div class="message-author"><?php echo htmlspecialchars($msg['name']); ?></div>  
 <p><?php echo nl2br(htmlspecialchars($msg['message'])); ?></p>  
 </div>  
<?php endforeach; ?>  
  
<hr>  
<form method="POST">  
 <label for="name"> </label>  
 <input type="text" id="name" name="name" placeholder="Имя">  
 <label for="message"> </label>  
 <textarea id="message" name="message" required placeholder="Ваше сообщение"></textarea>  
 <button type="submit">Отправить</button>  
</form>  
</body>  
</html>

**Init.php**

<?php  
try {  
 $db = new PDO('sqlite:Bookofguests.db');  
 $db->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  
  
 $db->exec("CREATE TABLE IF NOT EXISTS Bookofguests (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 name TEXT NOT NULL,  
 message TEXT NOT NULL,  
 created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP  
 )");  
 echo "БД 'Bookofguests' успешно создана ";  
} catch (PDOException $e) {  
 die("Не удалось подключиться к базе данных: " . $e->getMessage());  
}

**Задача 8:**

****

**Решение:**

**getcities.php**

<?php  
  
header('Content-Type: application/json');  
  
$username = 'cli';  
$password = '12344321';  
$apiUrl = 'http://exercise.develop.maximaster.ru/service/city';  
  
$ch = curl\_init();  
curl\_setopt($ch, *CURLOPT\_URL*, $apiUrl);  
curl\_setopt($ch, *CURLOPT\_RETURNTRANSFER*, true);  
curl\_setopt($ch, *CURLOPT\_HTTPAUTH*, *CURLAUTH\_BASIC*);  
curl\_setopt($ch, *CURLOPT\_USERPWD*, "$username:$password");  
  
$response = curl\_exec($ch);  
curl\_close($ch);  
  
echo $response;

**process.php**

<?php  
  
header('Content-Type: application/json');  
  
$username = 'cli';  
$password = '12344321';  
$city = isset($\_GET['city']) ? trim($\_GET['city']) : null;  
$weight = isset($\_GET['weight']) ? trim($\_GET['weight']) : null;  
  
if (empty($city) || empty($weight)) {  
 echo json\_encode(['error' => 'Вы не указали город и вес (сервер нельзя обманывать) .']);  
 exit;  
}  
  
if (!is\_numeric($weight) || $weight <= 0) {  
 echo json\_encode(['error' => 'Вам удалось обмануть клиентскую часть, но введите положительное число.']);  
 exit;  
}  
  
$apiUrl = 'http://exercise.develop.maximaster.ru/service/delivery';  
  
$options = [  
 'http' => [  
 'method' => 'GET',  
 'header' => [  
 'Authorization: Basic ' . base64\_encode("$username:$password"),  
 'Content-Type: application/json',  
 'Accept: application/json'  
 ],  
 'ignore\_errors' => true  
 ]  
];  
  
$context = stream\_context\_create($options);  
$response = file\_get\_contents($apiUrl . "?city=" . urlencode($city) . "&weight=" . urlencode($weight), false, $context);  
  
if ($response === FALSE) {  
 echo json\_encode(['error' => 'Ошибка при запросе к серверу Максимастер.']);  
} else {  
 echo $response;  
}

**CSS**

 .error {

            color: red;

        }

        select, input {

            width: 100%;

            padding: 10px;

            box-sizing: border-box;

            margin-bottom: 15px;

        }

        .form-container {

            display: flex;

            flex-direction: column;

            align-items: center;

            margin-left: 150px;

            margin-top: 50px;

            width: 300px;

        }

        button {

            align-self: center;

            width: 150px;

            padding: 10px 20px;

        }

**Html**

<!DOCTYPE html>

<head>

    <title>Калькулятор доставки</title>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="Calculateofdelievering.css">

</head>

<body>

<div class="form-container">

    <label for="city"> </label>

    <select id="city" name="city"></select>

    <input type="text" name="weight" placeholder="Вес, кг">

    <button type="button" id="calculateButton">Рассчитать</button>

    <div id="result"></div>

</div>

<script src="Calculateofdelievering.js">

</script>

</body>

**JS**

    const CACHE\_FILE = 'cities\_cache.json';

    function isToday(date) {

        const today = new Date();

        return date.getDate() === today.getDate() &&

            date.getMonth() === today.getMonth() &&

            date.getFullYear() === today.getFullYear();

    }

    function getFromCache() {

        try {

            const cacheData = localStorage.getItem(CACHE\_FILE);

            if (!cacheData) return null;

            const { data, timestamp } = JSON.parse(cacheData);

            const cacheDate = new Date(timestamp);

            if (isToday(cacheDate)) {

                return data;

            }

            return null;

        } catch (e) {

            console.error('Ошибка чтения кэша:', e);

            return null;

        }

    }

    function saveToCache(data) {

        try {

            const cacheData = {

                data: data,

                timestamp: new Date().toISOString()

            };

            localStorage.setItem(CACHE\_FILE, JSON.stringify(cacheData));

        } catch (e) {

            console.error('Ошибка сохранения кэша:', e);

        }

    }

    async function getCitiesfromMaximaster() {

        const cachedCities = getFromCache();

        if (cachedCities) {

            console.log('Используем данные из кэша');

            populateCitySelect(cachedCities);

            return;

        }

        try {

            const response = await fetch('getcities.php', { method: 'GET' });

            if (!response.ok) {

                throw new Error(`Ошибка HTTP: ${response.status}`);

            }

            const cities = await response.json();

            console.log('Получены данные с сервера:', cities);

            saveToCache(cities);

            populateCitySelect(cities);

        } catch (error) {

            console.error('Ошибка при получении данных:', error);

            document.getElementById('result').innerHTML = `<p class="error">Ошибка: ${error.message}</p>`;

        }

    }

    function populateCitySelect(cities) {

        const select = document.getElementById('city');

        select.innerHTML = '';

        cities.forEach(city => {

            const option = document.createElement('option');

            option.value = city;

            option.text = city;

            select.appendChild(option);

            if (city === 'Москва') {

                option.selected = true;

            }

        });

    }

    async function calculatedelievering() {

        const city = document.getElementById('city').value;

        let weight = document.querySelector('input[name="weight"]').value.trim();

        if (weight.startsWith("+")) {

            weight = weight.substring(1);

        }

        if (!city || !weight) {

            document.getElementById('result').innerHTML = '<p class="error">Пожалуйста, выберите город из списка и введите вес.</p>';

            return;

        }

        const weightNumber = parseFloat(weight);

        if (isNaN(weightNumber) || weightNumber <= 0) {

            document.getElementById('result').innerHTML = '<p class="error">Пожалуйста, введите корректное и положительное число (вес).</p>';

            return;

        }

        const apiUrl = `process.php?city=${encodeURIComponent(city)}&weight=${encodeURIComponent(weightNumber)}`;

        let attempts = 0;

        const maxAttempts = 3;

        while (attempts < maxAttempts) {

            attempts++;

            try {

                const response = await fetch(apiUrl, {

                    method: 'GET',

                    headers: {

                        'Content-Type': 'application/json'

                    }

                });

                if (!response.ok) {

                    throw new Error(`Ошибка HTTP: ${response.status}`);

                }

                const result = await response.json();

                if (result.status === "error" && result.price === 0) {

                    if (attempts < maxAttempts) {

                        console.log(`Попытка ${attempts} не удалась: ${result.message}. Повторяем запрос...`);

                        continue;

                    } else {

                        document.getElementById('result').innerHTML = `<p class="error">Ошибка: ${result.message}. Превышено количество попыток.</p>`;

                        return;

                    }

                }

                document.getElementById('result').innerHTML = `<p>Стоимость доставки в город ${city} груза весом ${weightNumber} кг равна ${result.price} руб.</p>`;

                return;

            } catch (error) {

                console.error('Ошибка при отправке данных:', error);

                document.getElementById('result').innerHTML = `<p class="error">Ошибка: ${error.message}</p>`;

                return;

            }

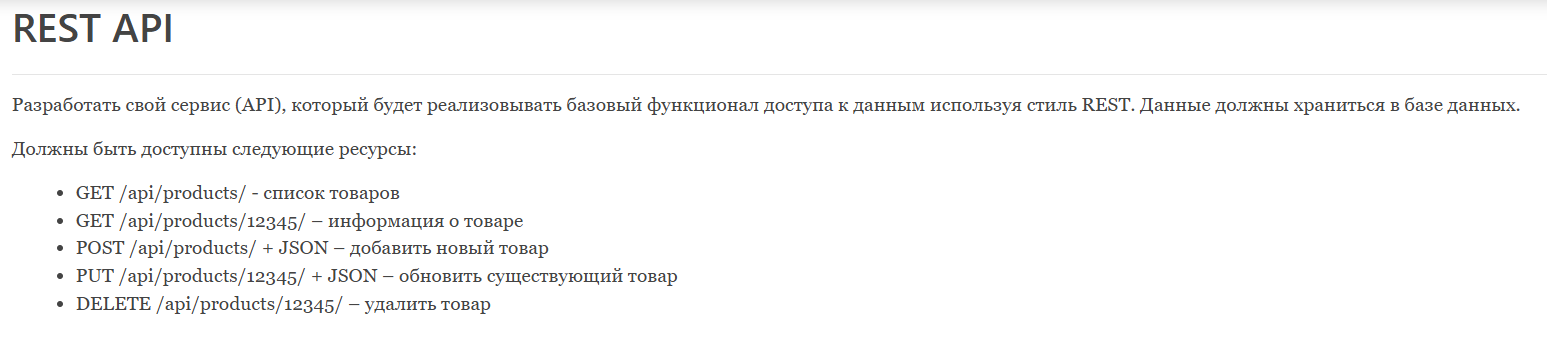
        }

    }

    document.getElementById('calculateButton').onclick = calculatedelievering;

    window.onload = getCitiesfromMaximaster;

**Задача 9:**

****

**Решение:**

**Index.php**

<?php  
require 'vendor/autoload.php';  
  
use Slim\Factory\AppFactory;  
use Slim\Exception\HttpNotFoundException;  
use Psr\Http\Message\ResponseInterface as Response;  
use Psr\Http\Message\ServerRequestInterface as Request;  
  
$app = AppFactory::*create*();  
$app->setBasePath('/simple-api');  
$app->addBodyParsingMiddleware();  
  
$db = new PDO('sqlite:products.db');  
$db->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  
  
$db->exec('CREATE TABLE IF NOT EXISTS products (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 name TEXT NOT NULL,  
 price REAL,  
 description TEXT,  
 created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP  
)');  
  
function jsonResponse(*Response* $response, $data, int $status = 200): *Response* {  
 $response->getBody()->write(json\_encode($data, *JSON\_UNESCAPED\_UNICODE*));  
 return $response->withHeader('Content-Type', 'application/json')->withStatus($status);  
}  
  
$app->get('/test', function (*Request* $request, *Response* $response): *Response* {  
 $response->getBody()->write("API работает!");  
 return $response;  
});  
  
$app->get('/api/products', function (*Request* $request, *Response* $response) use ($db): *Response* {  
 $stmt = $db->query('SELECT \* FROM products');  
 return jsonResponse($response, $stmt->fetchAll(PDO::*FETCH\_ASSOC*));  
});  
  
$app->get('/api/products/{id}', function (*Request* $request, *Response* $response, array $args) use ($db): *Response* {  
 $stmt = $db->prepare('SELECT \* FROM products WHERE id = ?');  
 $stmt->execute([$args['id']]);  
 $product = $stmt->fetch(PDO::*FETCH\_ASSOC*);  
  
 return $product  
 ? jsonResponse($response, $product)  
 : jsonResponse($response, ['error' => 'Товар не найден'], 404);  
});  
  
$app->post('/api/products', function (*Request* $request, *Response* $response) use ($db): *Response* {  
 $data = $request->getParsedBody();  
  
 if (empty($data['name']) || !isset($data['price'])) {  
 return jsonResponse($response, ['error' => 'Название и цена обязательны'], 400);  
 }  
  
 $stmt = $db->prepare('INSERT INTO products (name, price, description) VALUES (?, ?, ?)');  
 $stmt->execute([$data['name'], $data['price'], $data['description'] ?? null]);  
  
 return jsonResponse($response, ['id' => $db->lastInsertId()], 201);  
});  
  
$app->put('/api/products/{id}', function (*Request* $request, *Response* $response, array $args) use ($db): *Response* {  
 $data = $request->getParsedBody();  
  
 if (empty($data['name']) || !isset($data['price'])) {  
 return jsonResponse($response, ['error' => 'Название и цена обязательны'], 400);  
 }  
  
 $stmt = $db->prepare('UPDATE products SET name=?, price=?, description=? WHERE id=?');  
 $stmt->execute([$data['name'], $data['price'], $data['description'] ?? null, $args['id']]);  
  
 return $stmt->rowCount() > 0  
 ? jsonResponse($response, ['success' => true])  
 : jsonResponse($response, ['error' => 'Товар не найден'], 404);  
});  
  
$app->delete('/api/products/{id}', function (*Request* $request, *Response* $response, array $args) use ($db): *Response* {  
 $stmt = $db->prepare('DELETE FROM products WHERE id=?');  
 $stmt->execute([$args['id']]);  
  
 if ($stmt->rowCount() > 0) {  
 return jsonResponse($response, ['message' => 'Товар успешно удалён'], 200);  
 } else {  
 return jsonResponse($response, ['error' => 'Товар не найден'], 404);  
 }  
});  
  
$app->add(function (*Request* $request, $handler) {  
 try {  
 return $handler->handle($request);  
 } catch (HttpNotFoundException $e) {  
 $response = new \Slim\Psr7\Response();  
 $response->getBody()->write(json\_encode([  
 'error' => 'Маршрут не найден',  
 'available\_routes' => [  
 'GET /api/products',  
 'GET /api/products/{id}',  
 'POST /api/products',  
 'PUT /api/products/{id}',  
 'DELETE /api/products/{id}'  
 ]  
 ], *JSON\_UNESCAPED\_UNICODE*));  
 return $response->withStatus(404)->withHeader('Content-Type', 'application/json');  
 }  
});  
  
$app->run();

**.htaccess**

<IfModule mod\_rewrite.c>

    RewriteEngine On

    RewriteBase /simple-api/

    RewriteCond %{REQUEST\_FILENAME} !-f

    RewriteCond %{REQUEST\_FILENAME} !-d

    RewriteRule ^ index.php [QSA,L]

</IfModule>

**Composer.json**

{

    "require": {

        "slim/slim": "^4.0",

        "slim/psr7": "^1.6"

    }

}

# Краткие выводы студента по прохождению учебной практики

Производственная практика проводилась в ООО «Максимастер». При приеме на практику меня ознакомили с техникой безопасности на рабочем месте и дистанционным форматом проводимой практики. В рамках проводимой практики я улучшил свои знания в написании API, разобрался как работает AJAX, научился работать с LocalStorage, улучшил свои знания в языках программирования PHP и JavaScript.

Предприятием были обеспечены условия прохождения практики: обеспечена связь с руководителем практики, обеспечен график выполнения заданий, выданы материалы для самообучения, были предоставлены практические задания.

Руководителем практики осуществлялся контроль над выполнением программы практики ,оказывалась помощь, в случае необходимости, а также руководителем практики была написана характеристика.

Со стороны университета руководитель практики также предоставила все условия прохождения практики. Я доволен организацией практики как со стороны предприятия, так и университета.

# Сведения о ходе выполнения заданий студентом в процессе прохождения практики

|  |  |  |
| --- | --- | --- |
| **Дата** | **Вид деятельности** | **Используемое ПО** |
| 31.03.2025 | 1. Инструктаж по охране труда и технике безопасности на рабочем месте.  2. Знакомство с целями и задачами практики.  3. Знакомство с местом прохождения практики | Текстовый редактор |
| 01.04.2025-02.04.2025 | Выполнение задач в соответствии с заданием ТГПУ | Текстовый редактор |
| 03.04.2025 -06.04.2025 | Знакомство с объектами события и цветами в CSS. Выполнение задания на основе материала для самообучения | Visual Studio Code |
| 07.04.2025 -09.04.2025 | Создание интерфейса оформления заказа с использованием API Яндекс карт | Visual Studio Code |
| 10.04.2025-  12.04.2025 | Знакомство с форматом JSON и AJAX- запросами методом fetch. Создание таблицы товаров на основе материалов для самообучения | Visual Studio Code |
| 13.04.2025 –  15.04.2025 | Изучение библиотеки Chart.js.Создание графика процессора на её основе | Visual Studio Code |
| 16.04.2025 | Знакомство с LocalStorage, разработка электронной таблицы | Visual Studio Code |
| 17.04.2025 | Выполнение 1ого задания блока PHP.Счётчик хитов | PHP Storm, сервер Apache |
| 18.04.2025 –  19.04.2025 | Разработка базового варианта гостевой книги для сайта | PHP Storm, сервер Apache, компактная встраиваемая СУБД SQLite |
| 20.04.2025 – 21.05.2025 | Разработка интеграции сервиса калькулятор доставки на сайт | PHP Storm, сервер Apache, Visual Studio Code |
| 22.05.2025 –  24.05.2025 | Разработка своего собственного API c CRUD функционалом, используя стиль REST | PHP Storm, сервер Apache, Composer,компактная встраиваемая СУБД SQLite |
| 25.04.2025-27.04.2025 | 1. Подготовка отчетной документации по практике.  2. Подведение итогов практики | Текстовый редактор |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Должность Подпись руководителя от организации

*(печать организации)*