Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Web-технологии”

Лабораторная работа №4

“Исследование возможностей асинхронного

взаимодействия с сервером. Технология AJAX”

Выполнил:

ст.гр. ИС/б-17-2

Долженко И.А.

Проверил:

Овчинников А.Л.

Севастополь

2020

1 ЦЕЛЬ РАБОТЫ

Изучить возможности асинхронного обмена данными с сервером, приобрести практические навыки использования технологии AJAX для организации отправки данных на сервер и получения данных без полной перезагрузки страницы.

2 ПОСТАНОВКА ЗАДАЧИ

1. На странице «Регистрация пользователя» реализовать проверку занятости логина без перезагрузки страницы.

2. Реализовать возможность комментирования записей блога для авторизированных пользователей сайта. В случае, если пользователь авторизирован отображать после каждой записи блога кнопку или ссылку «Добавить комментарий». При нажатии кнопки/ссылки отображать окно, содержащее поле ввода текста комментария и кнопку «Отправить». Реализовать отправку комментария на сервер и его отображение под записью блога без полной перезагрузки страницы. Комментарии хранить в специально разработанной таблице базы данных.

3. В зоне администратора реализовать возможность редактирования записей блога. Отображать возле каждой записи блога кнопку или ссылку «Изменить». При нажатии кнопки/ссылки отображать окно, содержащее поле ввода с темой записи блога и поле ввода с текстом записи блога для редактирования, а также кнопку/ссылку «Сохранить изменения». Реализовать отправку данных и их отображение без полной перезагрузки страницы.

Вариант 3

Задание 2 – Script, JSON

Задание 3 – XMLHttpRequest, XML

3 ХОД РАБОТЫ

1. На странице «Регистрация пользователя» реализуем проверку занятости логина без перезагрузки страницы.

Для этого реализуем функцию fetchCheck:

const resultBlock = document.getElementsByClassName("result-block")[0];

const loginInput = document.querySelector("input[name=login]");

const emailInput = document.querySelector("input[name=Email]");

const fetchCheck = (login, email) => {

    const formData = new FormData();

    formData.append("login", login);

    formData.append("email", email);

    fetch("checkLogin", { method: "POST", body: formData })

        .then(response => response.clone().json())

        .then(data => {

            resultBlock.innerHTML = "";

*if* (data.length) {

                data.forEach(err => {

                    const errorDiv = document.createElement("p");

                    errorDiv.className = "result-block\_\_item error";

                    errorDiv.innerHTML = err;

                    resultBlock.appendChild(errorDiv);

                });

            }

        });

};

loginInput.addEventListener("input", e => {

    fetchCheck(e.currentTarget.value, emailInput.value);

});

emailInput.addEventListener("input", e => {

    fetchCheck(loginInput.value, e.currentTarget.value);

});

И экшен из AuthController:

    function checkLoginAction() {

        $findUserByLogin = *$this*->model->findByField($\_POST['login'], 'login');

        $findUserByEmail = *$this*->model->findByField($\_POST['email'], 'email');

        $result = [];

*if* ($findUserByEmail != null) {

            array\_push($result, "Данный E-mail уже существует");

        }

*if* ($findUserByLogin != null) {

            array\_push($result, "Данный логин уже существует");

        }

        echo json\_encode($result);

    }

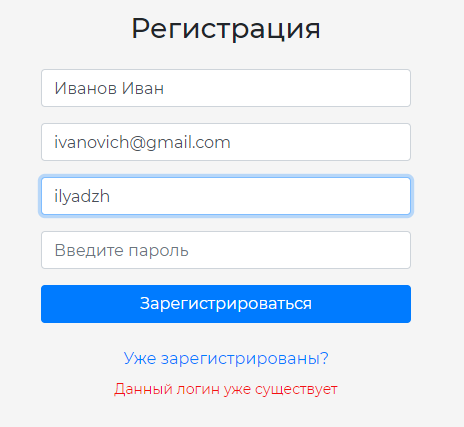


Рисунок 1 – Результат проверки занятности логина и email

2. Реализуем возможность комментирования записей блога для авторизированных пользователей сайта. В случае, если пользователь авторизирован будем отображать после каждой записи кнопку «Добавить комментарий» и поле для ввода комментария. Реализуем отправку комментария на сервер и его отображение под записью блога без полной перезагрузки страницы. Комментарии будем хранить в специально разработанной таблице базы данных.

JS функции, реализующие запрос и добавление нового комментария:

function createScript(id, fullname) {

    const input = document.querySelector(`.form-control[data-id='${id}']`);

*if* (input.value === "") *return*;

    const newScript = document.createElement("script");

    const date = new Date().format("yyyy-MM-dd h:mm:ss");

    newScript.src =

        "/blog/add/?id\_post=" +

        id +

        "&fullname=" +

        fullname +

        "&comment=" +

        input.value +

        "&date=" +

        date;

    document.getElementsByTagName("body")[0].appendChild(newScript);

}

function addComment(data) {

    const input = document.querySelector(`.form-control[data-id='${data.id}']`);

    const numberComments = input.parentNode.parentNode.parentNode.querySelector(

        ".card-comment\_\_title"

    );

    numberComments.innerHTML = parseInt(numberComments.innerHTML) + 1 + " Комментариев";

    const commentContainer = input.parentNode.parentNode.parentNode.querySelector(

        ".card-comment\_\_container"

    );

    commentContainer.insertAdjacentHTML(

        "afterbegin",

        `<div class="comment-item"><div class="d-flex"><div class="comment-item\_\_name">${data.fullname},</div><div class="comment-item\_\_date">${data.date}</div></div><div class="comment-item\_\_text">${data.comment}</div></div>`

    );

    input.value = "";

}

И экшен из BlogController:

    function addAction() {

*$this*->model->addComment($\_GET);

        $json = [

            'id' => $\_GET['id\_post'],

            'fullname' => $\_GET['fullname'],

            'comment' => $\_GET['comment'],

            'date' => $\_GET['date']

        ];

        echo "addComment(".json\_encode($json).");";

    }

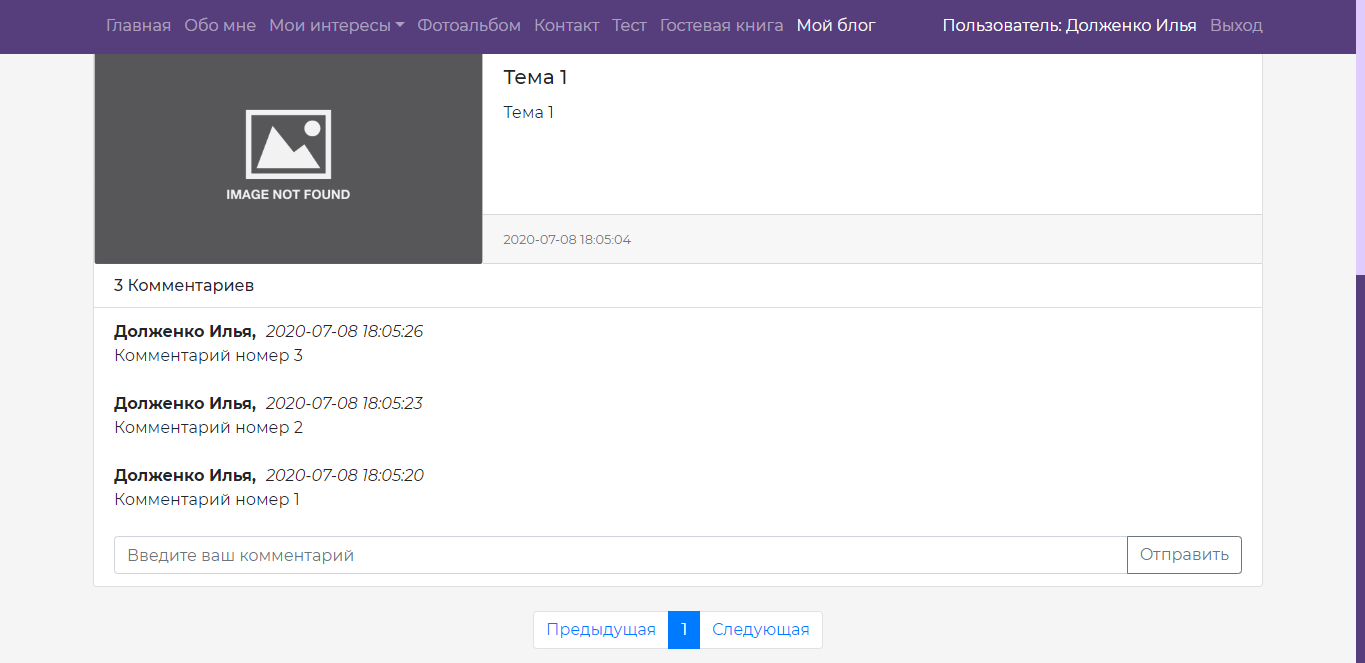


Рисунок 2 – Процесс добавления комментариев

3. В зоне администратора реализуем возможность редактирования записей блога. Будем отображать возле каждой записи блога кнопку «Изменить». При нажатии кнопки будем отображать окно, содержащее поле ввода с темой записи блога и поле ввода с текстом записи блога для редактирования, а также кнопку «Сохранить». Реализуем отправку данных и их отображение без полной перезагрузки страницы.

Файл editBlog.js:

const editBtn = document.getElementsByClassName("edit-btn");

const modal = document.querySelector(".modal");

const container = modal.querySelector(".container");

const modalTitle = document.getElementById("editTitle");

const modalMessage = document.getElementById("editMessage");

const modalErrorBlock = document.getElementById("modalErrorBlock");

let currentPost = {};

modal.addEventListener("click", e => {

*if* (e.target !== modal && e.target !== container) *return*;

    modal.classList.add("hidden");

});

*for* (let i = 0; i < editBtn.length; i++) {

    editBtn[i].addEventListener("click", e => {

        currentPost = {

            id: e.currentTarget.getAttribute("data-id"),

            title: e.currentTarget.getAttribute("data-title"),

            message: e.currentTarget.getAttribute("data-message")

        };

        modalTitle.value = currentPost.title;

        modalMessage.value = currentPost.message;

        modal.classList.remove("hidden");

        document.getElementById("saveBtn").addEventListener("click", saveChanges);

    });

}

const saveChanges = () => {

    const xmlString = "<profile>" +

    "  <id>" + currentPost.id + "</id>" +

    "  <title>" + modalTitle.value + "</title>" +

    "  <message>" + modalMessage.value + "</message>" +

    "</profile>";

    let xmlhttp = new XMLHttpRequest();

    xmlhttp.open("POST", "http://web/admin/editblog/edit", true);

    xmlhttp.setRequestHeader("Content-Type", "text/xml; charset=UTF-8");

    xmlhttp.send(xmlString);

    xmlhttp.onreadystatechange = () => {

*if* (xmlhttp.readyState == 4) {

*if* (xmlhttp.status == 200) {

                const data = JSON.parse(xmlhttp.response);

                modalErrorBlock.innerHTML = "";

*if* (data.length) {

                    data.forEach(err => {

                        const errorDiv = document.createElement("p");

                        errorDiv.className = "result-block\_\_item error";

                        errorDiv.innerHTML = err;

                        modalErrorBlock.appendChild(errorDiv);

                    });

                } *else* {

                    const currentBtn = document.querySelectorAll(

                        `.btn.edit-btn[data-id="${currentPost.id}"]`

                    )[0];

                    const title = currentBtn.parentNode.getElementsByTagName(

                        "h5"

                    )[0];

                    const text = currentBtn.parentNode.getElementsByTagName("p")[0];

                    title.innerHTML = modalTitle.value;

                    text.innerHTML = modalMessage.value;

                    currentBtn.setAttribute("data-title", modalTitle.value);

                    currentBtn.setAttribute("data-message", modalMessage.value);

                    modal.classList.add("hidden");

                }

            }

        }

    };

};

И экшен из AdminEditBlogController:

    function editAction() {

*$this*->checkIsAuth();

        $xml = simplexml\_load\_string(file\_get\_contents('php://input'));

        $json = json\_encode($xml);

        $array = json\_decode($json, TRUE);

*$this*->model->validator->validate($array);

        $errors = *$this*->model->validator->getErrors();

*if* (empty($errors)) {

*$this*->model->editPost($array);

        }

        echo json\_encode($errors);

    }

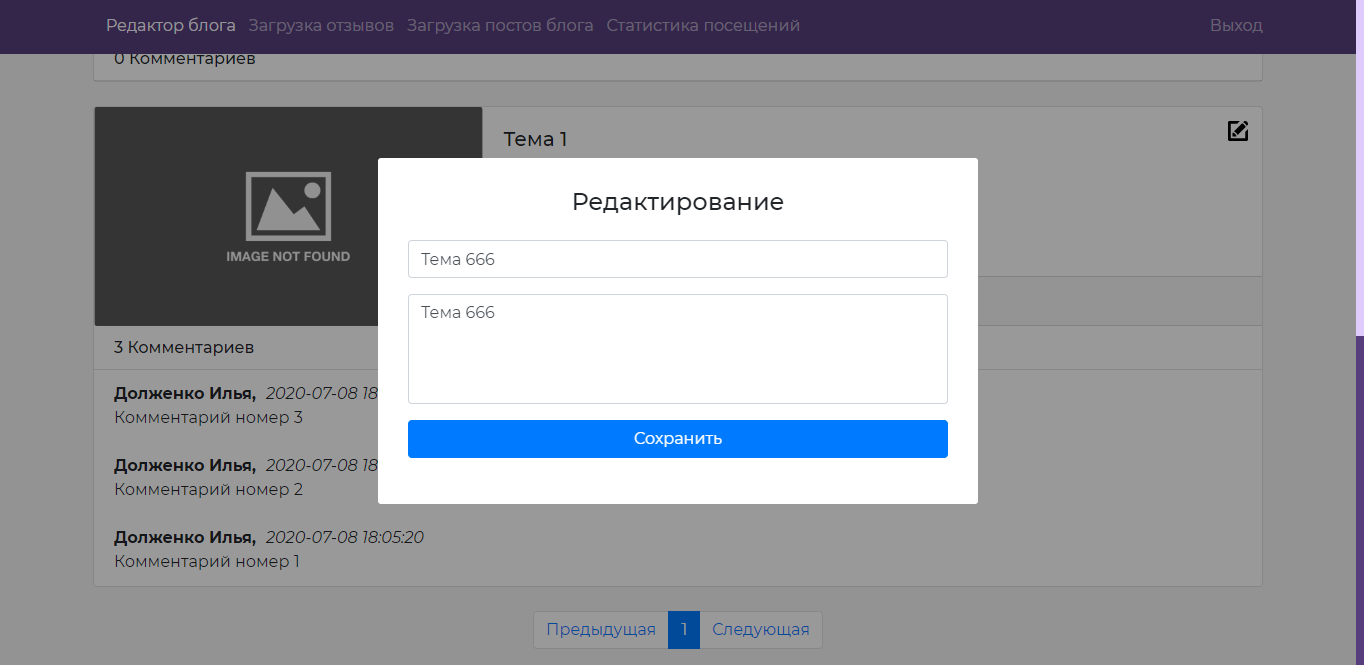


Рисунок 3 – Процесс редактирования темы

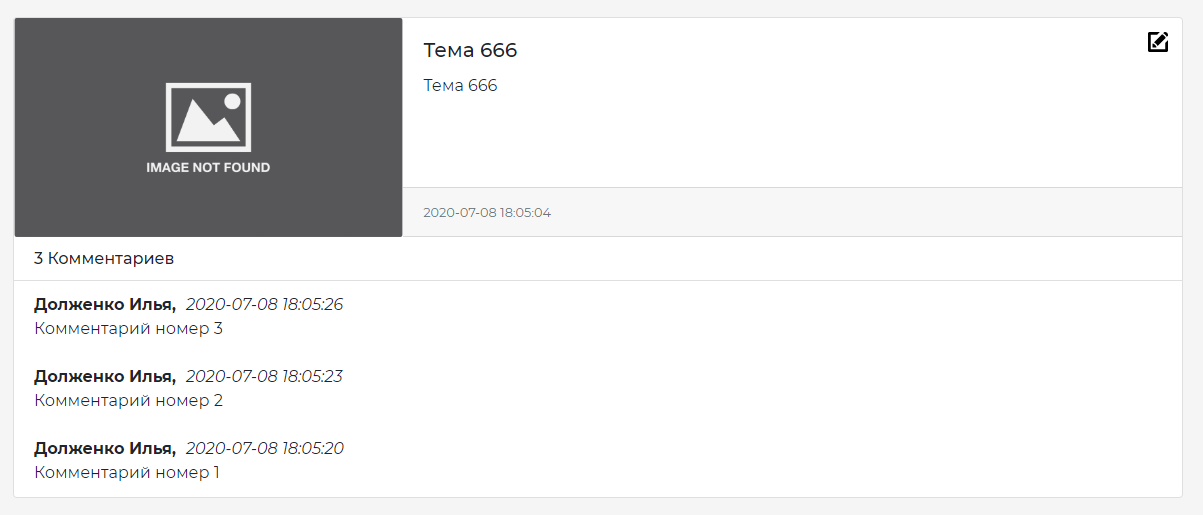


Рисунок 4 – Результат редактирования темы

ВЫВОД

В ходе выполнения лабораторной работы были изучены возможности асинхронного обмена данными с сервером, приобретены практические навыки использования технологии AJAX для организации отправки данных на сервер и получения данных без полной перезагрузки страницы.