Министерство науки и высшего образования Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: «Веб-технологии»

Лабораторная работа №11

«Исследование возможностей асинхронного взаимодействия  
с сервером. Технология AJAX.»

Выполнил

ст. гр. ИС/б-31-о

Носенко В.С.

Проверил:

Овчинников А.Л.

Севастополь

2020

1 ЦЕЛЬ

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 (использовать событие onBlur или кнопку «Проверить занятость»). Использовать для отправки данных на сервер метод и формат данных, указанный в колонке «Задание №1» таблицы 1.

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 модальное окно (реализовать с использванием, например, с использованием абсолютно спозиционированного блока div), содержащее поле ввода текста комментария и кнопку «Отправить». Реализовать отправку комментария на сервер и его отображение (дата и время, автор, текст комментария) под записью блога (или под уже существующими комментариями данной записи) без полной перезагрузки страницы. Комментарии хранить в специально разработанной таблице базы данных. Использовать для отправки данных на сервер метод и формат данных, указанный в колонке «Задание №2» таблицы 1.

3. В зоне администратора реализовать возможность редактирования записей блога. Отображать возле каждой записи блога кнопку или ссылку «Изменить». При нажатии кнопки/ссылки отображать окно (div), 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. Использовать для отправки данных на сервер метод и формат данных, указанный в колонке «Задание №3» таблицы 1.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Задание №1** | **Задание №2** | **Задание №3** |
| **9** | XMLHttpRequest, JSON | Fetch, XML | iFrame, Plaintext |

3 ХОД РАБОТЫ

1. Листинги разработанных программ.

Реализуем проверку занятости логина на странице «Регистрация пользователя». Используем для отправки данных на сервер метод XMLHttpRequest и формат данных JSON.

Файл CheckLogin.js:

**function** *greenForm*(){  
 **$**(**'#login'**).css(**"border"**, **"7px solid #79c603"**);  
}  
**function** *redForm*(){  
 **$**(**'#login'**).css(**"border"**, **"7px solid red"**);  
}  
  
**function** *CheckLogin*(){  
 **let** login = **$**(**'#login'**).val();  
  
 **let** xhr = **new *XMLHttpRequest***();  
  
 xhr.open(**'POST'**, **'http://web/registration/checkLogin'**);  
  
 xhr.setRequestHeader(**'Content-Type'**, **'application/json'**);  
  
 xhr.**onreadystatechange** = **function**(){  
 **if**(xhr.**readyState** === 4 && xhr.**status** === 200) {  
 **let** serverResponse = xhr.**responseText**;  
 **if**(serverResponse == 0){  
 *greenForm*();

**$**(**'#response'**).css(**"color"**, **"#79c603"**);  
**$**(**'#response'**).css(**"margin-left"**, **"15px"**);  
**$**(**'#response'**).text(***loginTrue***);

}**else**{  
 *redForm*();

**$**(**'#response'**).css(**"color"**, **"red"**);  
**$**(**'#response'**).css(**"margin-left"**, **"15px"**);  
**$**(**'#response'**).text(***loginFalse***);

}  
 }  
 };  
  
 xhr.send(***JSON***.stringify({**'login'**:login}));  
}

Обработка данных на сервере в контроллере RegistrationController.php:

**public function** checkLogin()  
{  
 *header*(**"Access-Control-Allow-Origin: \*"**);  
 $var = *json\_decode*(*file\_get\_contents*(**'php://input'**));  
 $user = **new** User();  
 **if** ($user->findLogin($var->**login**) == 0) {  
 **echo** 0;  
 } **else echo** 1;  
}

Создание кнопки «Проверить занятость» на view Registration.php:

<**div class = "form-buttons"**>  
 <**input type="button" id="buttonCheckLogin" class = "helper-button" value = "Проверить занятость" onclick="***CheckLogin*()**"**>  
</**div**>

<**p class="test" id="response"**></**p**>

Реализуем возможность добавления комментариев на странице «Мой блог». Отправлять данные на сервер будем с помощью метода fetch(), формат данных – xml.

Файл AddComment.js:

**let *postId*** = **""**;  
**$**(**function** () {  
 **$**(**".addComment"**).on(**'click'**, **function** () {  
 *OpenModalWindow*();  
 **let** id = **$**(**this**).val();  
 ***postId*** = id;  
 })  
});  
**function** *CloseModalWindow*() {  
 **$**(**".expanded"**).css(**"display"**, **"none"**);  
 **$**(**".test"**).css(**"filter"**, **"none"**);  
 **$**(**".test"**).css(**"pointer-events"**, **"auto"**);  
 **$**(**"body"**).css(**"background"**, **""**);  
}

**function** *OpenModalWindow*() {  
 **$**(**".expanded"**).css(**"display"**, **"block"**);  
 **$**(**".test"**).css(**"filter"**, **"blur(8px)"**);  
 **$**(**".test"**).css(**"pointer-events"**, **"none"**);  
 **$**(**"body"**).css(**"background"**, **"#000"**);  
}  
**async function** *AddComment*() {  
 **let** comment = **$**(**'#comment'**).val();  
 **let** author = **$**(**'#userName'**).text();  
 **let** xmlString = **"<comment>"** +  
 **" <idBlog>"** + *escape*(***postId***) + **"</idBlog>"** +  
 **" <commentText>"** + *escape*(comment) + **"</commentText>"** +  
 **"</comment>"**;  
 **let** res = **await** *fetch*(**"/blog/addComment"**, {  
 **method**: **'post'**,  
 **headers**: {  
 **"Content-type"**: **"text/xml"** },  
 **body**: xmlString  
 })  
 .then(response => response.text())  
 .then(body => {  
 ***console***.log(body);  
 **$**(**'#comment'**+ ***postId***).append(**'<p><b>Автор: </b><i>'** + author + **'</i><br>'** + comment + **'</p>'**);  
 *CloseModalWindow*();  
 })  
}

Обработка данных на сервере в контроллере MyBlogController.php:

**public function** addComment()  
{  
 *session\_start*();  
  
 *header*(**"Access-Control-Allow-Origin: \*"**);  
 $var = **new** SimpleXMLElement(*file\_get\_contents*(**'php://input'**));  
 $commentArray = (**array**)$var;  
  
 **if**(!**empty**($commentArray)){  
 $comment = **new** Comment();  
  
 $comment->**author** = $\_SESSION[**'userId'**];  
 $comment->**dateTime** = $comment->setNowDate();  
 $comment->**comment** = $commentArray[**'commentText'**];  
 $comment->**postId** = $commentArray[**'idBlog'**];  
  
 $comment->save();  
  
 **echo** 1;  
 }  
}

Код модального окна, кнопки и отображения комментариев на странице view MyBlog.php:

Модальное окно:

<**div class="expanded"**>  
 <**div class="modalwindow"**>  
 <**h3**>Добавить коментарий!</**h3**>  
 <**form onsubmit="return false**;**"**>  
 <**textarea name="comment" id="comment" placeholder="Введите ваш коментарий..." cols="40"  
 rows="10"**></**textarea**><**br**>  
 <**button class="asyncButton" type="reset" onclick="***CloseModalWindow*();**"**>Закрыть</**button**>  
 <**button class="asyncButton" type="submit" onclick="***AddComment*();**" id="addB"**>Добавить</**button**>  
 </**form**>  
 </**div**>  
</**div**>

Кнопка и вывод комментариев:

**<?php if** ($\_SESSION[**'isUser'**]) { **?>** <**li**>  
 <**button style="background**: **#c5b7db**; **font-size**: 28**px" class="addComment" value="<?=** $post[**'id'**]; **?>"**>Добавить комментарий</**button**>  
 </**li**>  
**<?php** } **?>**<**li**>  
 **<?php**$comment = **new** Comment();  
$comments = $comment->findUserComments($post[**'id'**]);  
**if**($comments != 0) { **?>**<**div style="border**: 7**px solid #ac8dc9**;**" class="commentDiv" id="comment<?=** $post[**'id'**]; **?>"**>  
 **<?php foreach** ($comments **as** $comment){  
 **echo '<p><b>Автор: </b><i>'** . $comment[**'fio'**] . **'</i><br>'** . $comment[**'comment'**] . **'</p>'**;  
 } **?>**</**div**>

**<?php** }**else**{ **?>** <**div class="commentDiv" id="comment<?=** $post[**'id'**]; **?>"**>  
 **<?php** }  
 **?>**</**li**>

Запрос к БД findUserComments():

**public static** *$selectQueryByUserComments* = **'SELECT %s FROM `%s` INNER JOIN %s ON %s WHERE postId=?'**;

Обработка запроса:

**public function** findUserComments($postId)  
{  
 $on = **"author = users.id"**;  
 $query = *sprintf*(  
 SqlBuilder::*$selectQueryByUserComments*,  
 **"\*"**,  
 **static**::*$table*,  
 **"users"**,  
 $on  
 );  
 $result = **parent**::*execute*($query, [$postId], **self**::***many\_type***);  
  
 **if** (!$result) {  
 **return** 0;  
 }  
 **return** $result;  
}

Реализуем на странице «Редактор блога» в зоне администратора сайта возможность редактирования постов. Метод для отправки данных на сервер в данном случае будет осуществляться с помощью динамического iframe, формат данных – PlainText.

Файл EditPost.js:

**let *postId*** = **""**;  
**let *myForm*** = **$**(**'#editForm'**);  
**let *iframe***;  
  
**$**(**function** () {  
 **$**(**".editPostButton"**).on(**'click'**, **function** () {  
 *OpenModalWindow*();  
 ***postId*** = **$**(**this**).val();  
 ***console***.log(***postId***);  
 **let** themeText = **$**(**'#themeText'** + ***postId***).text();  
 **$**(**'#editTheme'**).val(themeText);  
 **let** postText = **$**(**'#postText'** + ***postId***).text();  
 **$**(**'#editPostText'**).val(postText);  
 **$**(**'#editId'**).val(***postId***);  
 })  
});  
  
**$**(**function** () {  
 **$**(**'#saveUpdatePost'**).click(**function** () {  
 **let** newTheme = **$**(**'#editTheme'**).val();  
 **let** newPostText = **$**(**'#editPostText'**).val();  
  
 ***iframe*** = ***document***.createElement(**'iframe'**);  
 ***iframe***.**name** = **"myFrame"**;  
  
 ***iframe***.**onload** = **function** () {  
 **if**(**this**.**contentWindow**.**document**.**body**.**innerHTML**){  
 **$**(**'#themeText'** + ***postId***).text(newTheme);  
 **$**(**'#postText'** + ***postId***).text(newPostText);  
 *CloseModalWindow*();  
 **$**(***iframe***).remove();  
 }  
 };  
  
 ***document***.**body**.appendChild(***iframe***);  
 ***myForm***.submit();  
 });  
});

**function** *CloseModalWindow*() {  
 **$**(**".expanded"**).css(**"display"**, **"none"**);  
 **$**(**".test"**).css(**"filter"**, **"none"**);  
 **$**(**".test"**).css(**"pointer-events"**, **"auto"**);  
 **$**(**"body"**).css(**"background"**, **""**);  
}  
  
**function** *OpenModalWindow*() {  
 **$**(**".expanded"**).css(**"display"**, **"block"**);  
 **$**(**".test"**).css(**"filter"**, **"blur(8px)"**);  
 **$**(**".test"**).css(**"pointer-events"**, **"none"**);  
 **$**(**"body"**).css(**"background"**, **"#000"**);  
}

Обработка данных на сервере в контроллере AdminController.php:

**public function** updatePost()  
{  
 $post = **new** Post();  
  
 $post->setId($\_POST[**'editId'**]);  
 $post->setTheme($\_POST[**'editTheme'**]);  
 $post->setMessage($\_POST[**'editPostText'**]);  
  
 $post->update();  
  
 **echo** 1;  
}

Код кнопки на view BlogEditor.php:

<**li**>  
 <**button type="button" style="background**: **#c5b7db**; **font-size**: 28**px" class="editPostButton" value="<?=** $post[**'id'**]; **?>"**>Изменить</**button**>  
</**li**>

1. Внешний вид разработанных Web-страниц.

Проверка занятости пароля (см. рисунок 1a, 1б):

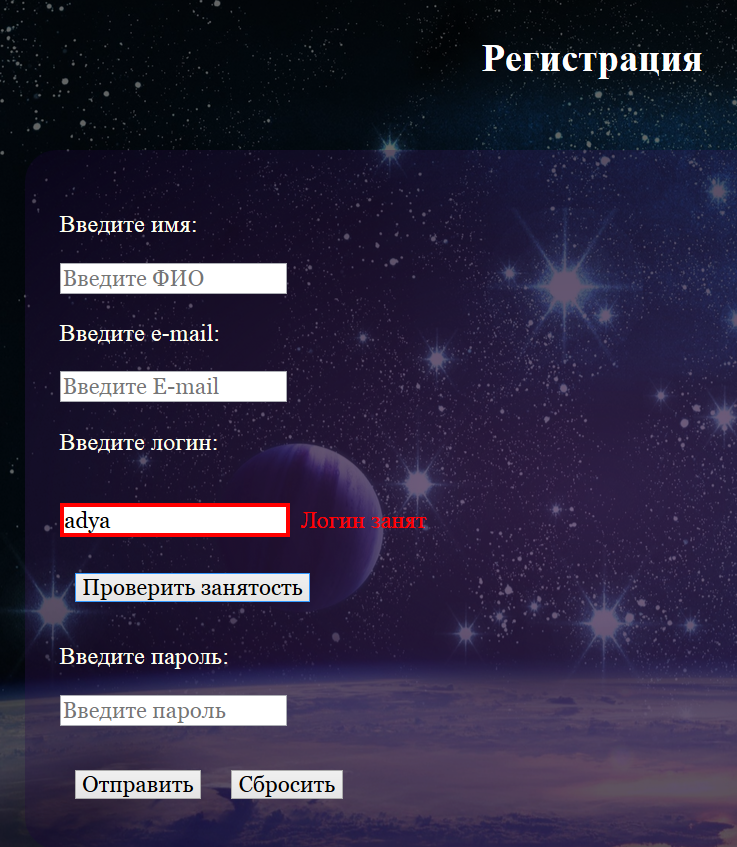
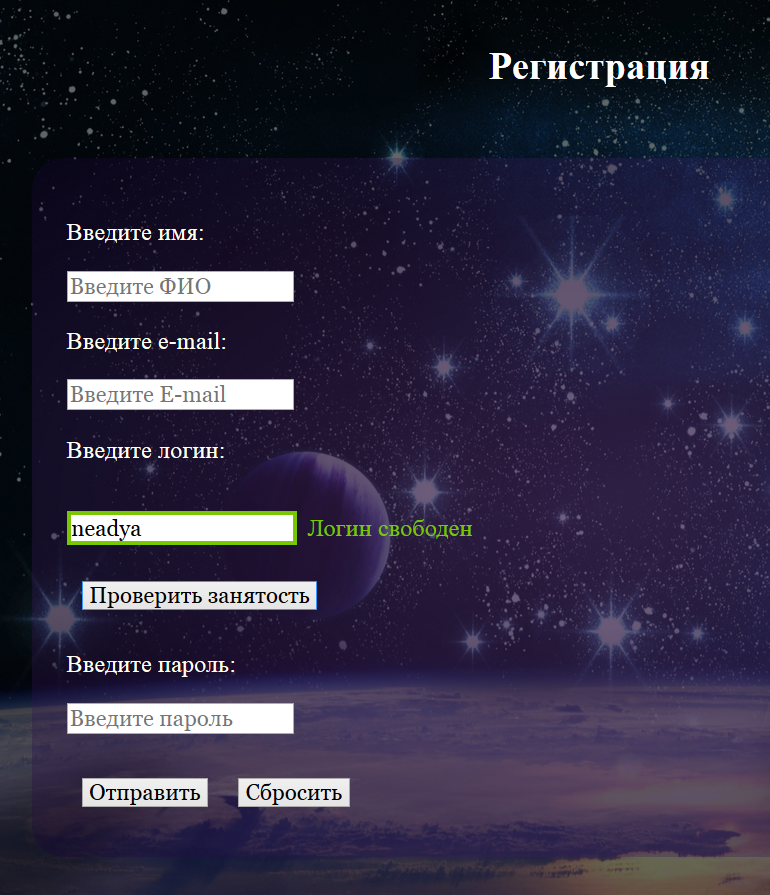
 

Рисунок 1 – Проверка занятости логина

а) Логин занят;

б) Логин свободен

На странице «Мой блог» появилась кнопка «Добавить комментарий» перед ранее добавленными комментариями к конкретному посту (см. рисунок 2):



Рисунок 2 – Кнопка «Добавить комментарий»

На рисунке 3 продемонстрировано модальное окно для добавления комментария:

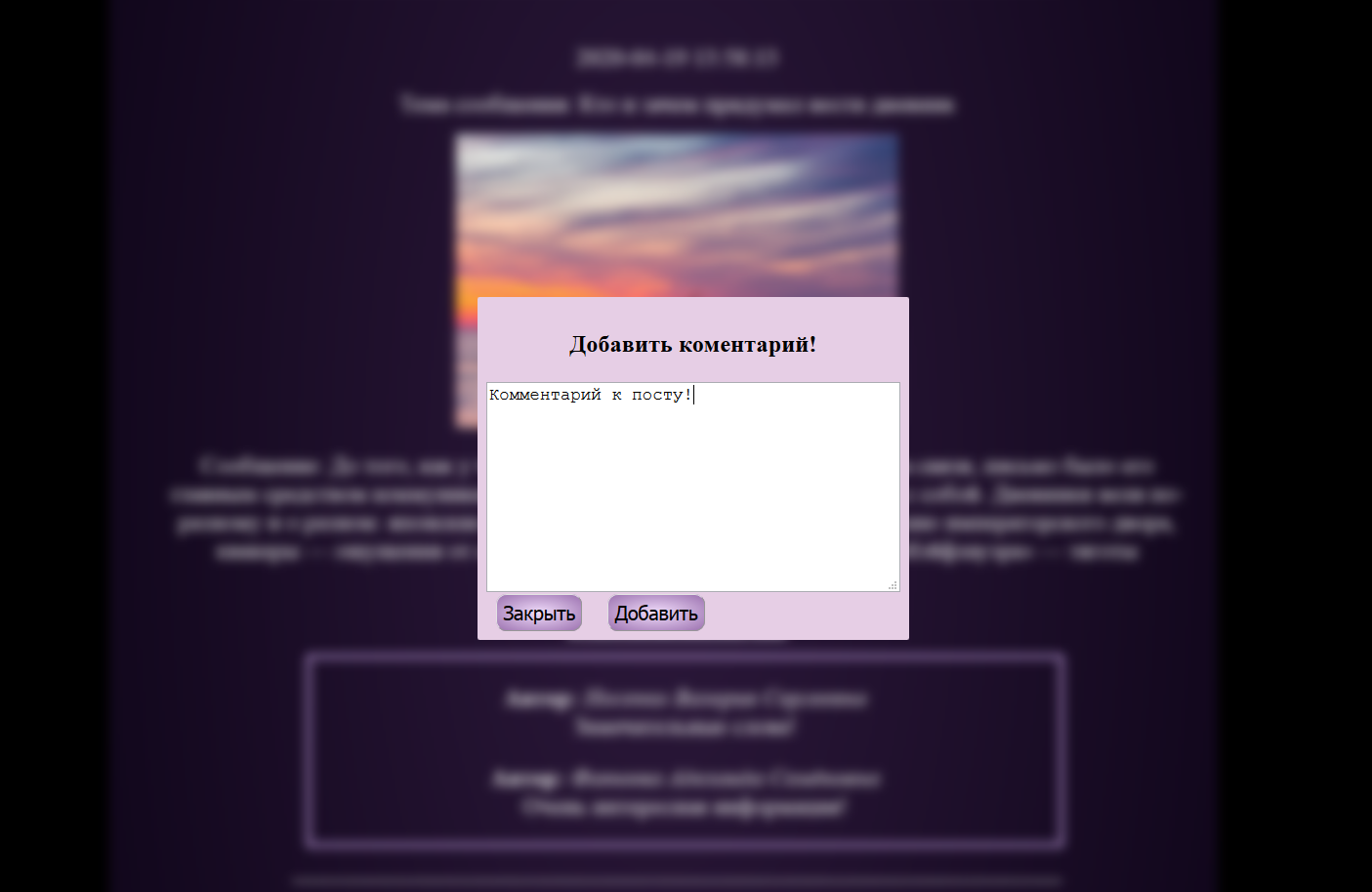


Рисунок 3 – Модальное окно добавления комментария

На рисунке 4 видно, что комментарий добавился ниже существующих.



Рисунок 4 – Проверка добавленного комментария на существование

Разработанная таблица базы данных для хранения пользовательских комментариев (см. рисунок 5):

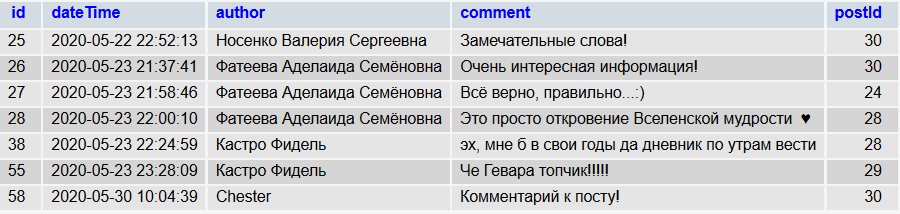


Рисунок 5 – Таблица базы данных «commwnts»

На странице «Редактор блога» появилась кнопка «Изменить» содержимое поста (см. рисунок 6):

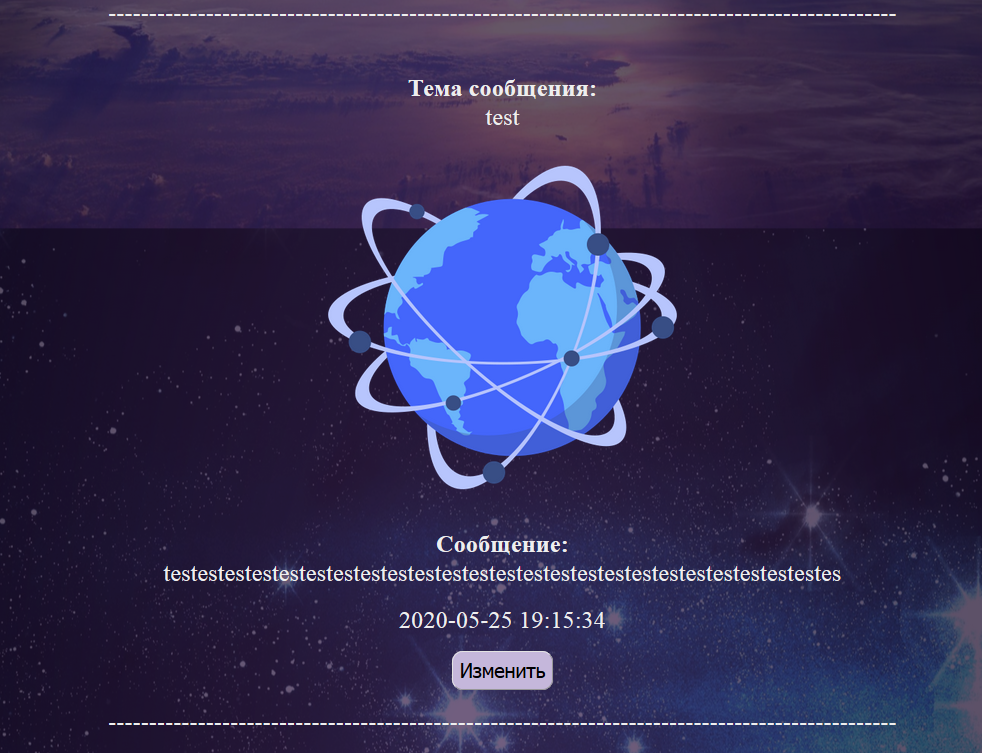


Рисунок 6 – Кнопка «Изменить»

Модальное окно для изменения содержимого поста (см. рисунок 7):

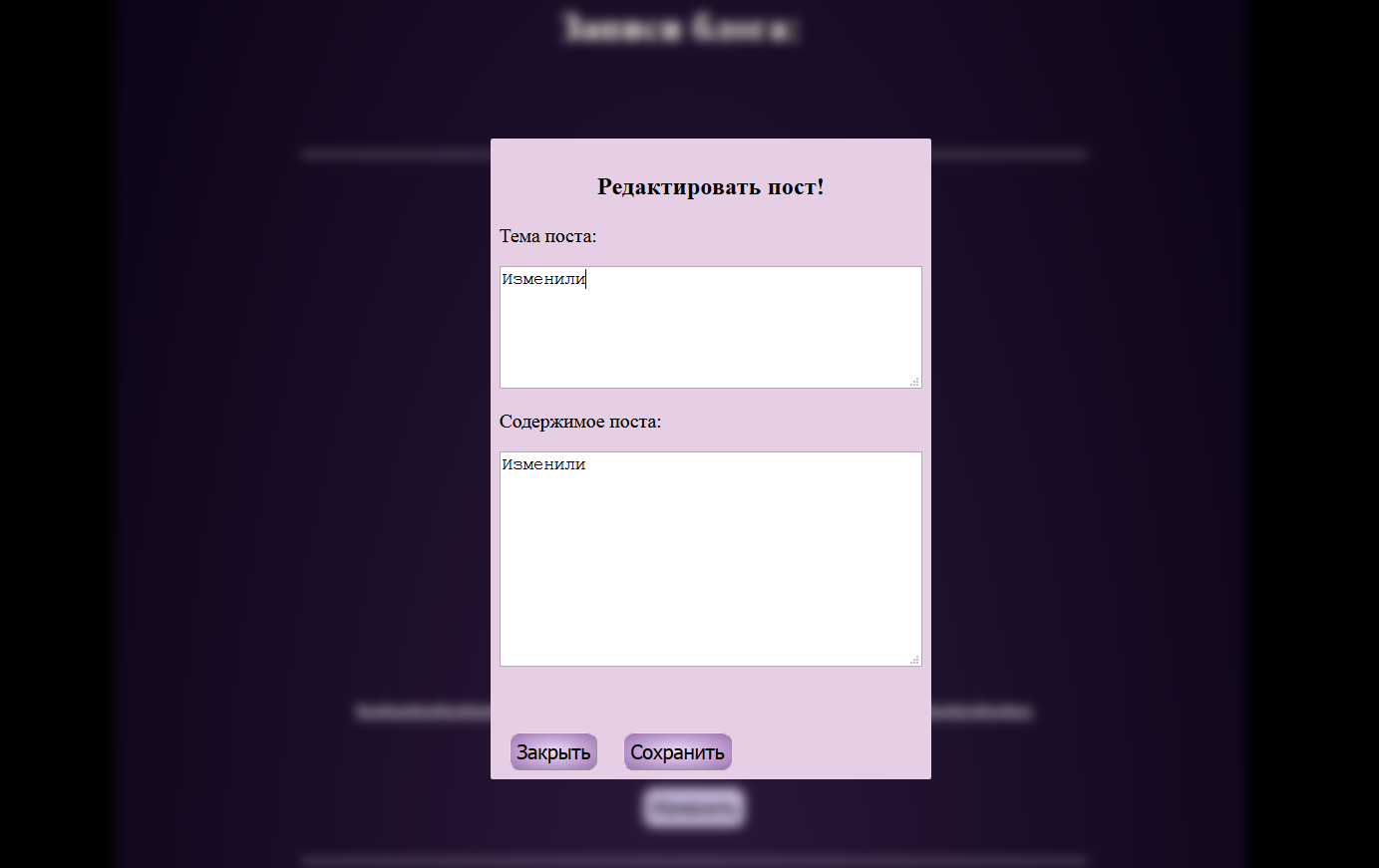


Рисунок 7 – Модальное окно для изменения содержимого поста

Изменённые данные обновились без перезагрузки страницы (см. рисунок 8):

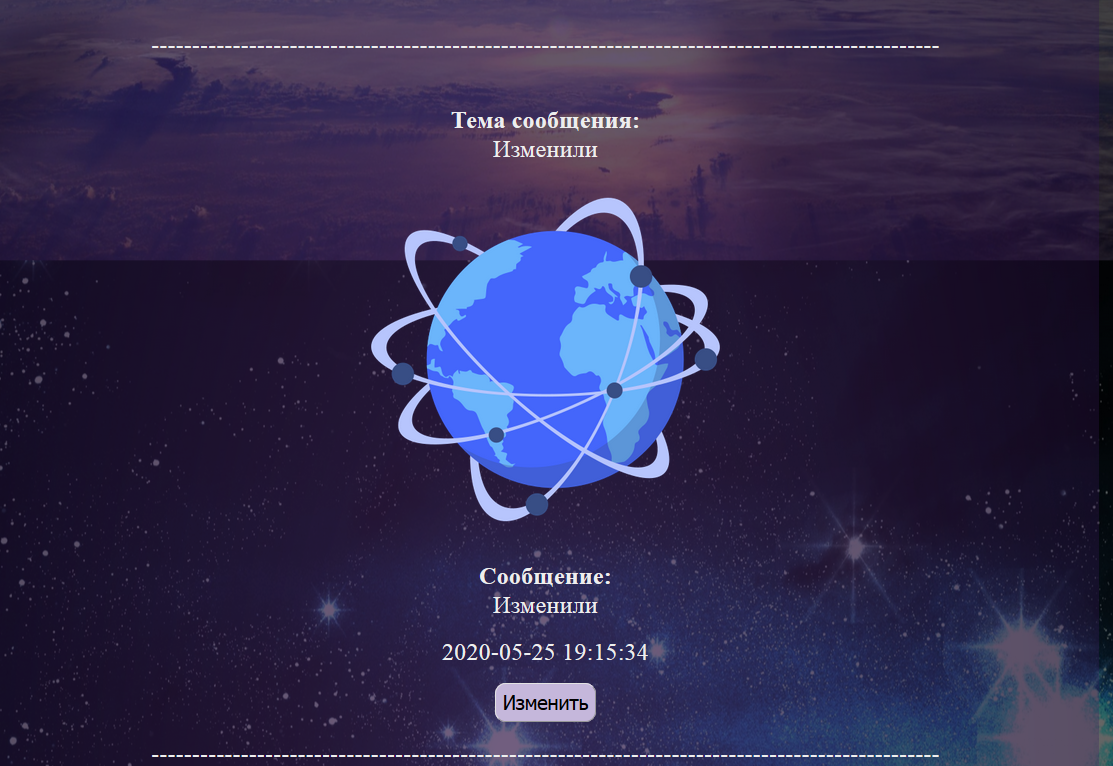


Рисунок 8 – Обновлённый пост

Данные в таблице «posts» БД, содержащей посты также изменились:



Рисунок 9 – Обновлённый пост в БД

ВЫВОДЫ

В ходе данной 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.