Комитет по науке и высшей школе

Политехнический колледж городского хозяйства

ПРАКТИКУМ

по разделам «Программирование на *php* и *mysql*»

Санкт-Петербург

2020

**Автор:** Леонид Владимирович Ильюшенков – кандидат технических наук, преподаватель высшей категории

Людмила Вульфовна Левит — Председатель цикловой комиссии специальности 09.02.07, преподаватель высшей категории

**Рецензент:**

Л.В. Ильюшенков Л.В. Левит

**ПРАКТИКУМ по разделам «Программирование на php и mysql».**  Практикум предназначен для выполнения практических работ по профессиональным модулям специальности 09.02.07 – Информационные технологии и программирование всех форм обучения. СПБ: Политехнический колледж городского хозяйства, 2020 – с.

Практикум предназначен для студентов среднего профессионального образования. Особое внимание уделено применению локального веб-сервера для Windows - Open Server. Рассмотрены вопросы отладки php кода, работа с базой данных mysql, регистрации и авторизации пользователей, администрирования сайта. Даны сведения о работе с электронной почтой. В конце главы даны вопросы самостоятельной работы.

Введение

**Темы для проектирования сайтов**

Для выполнения практических работ Вам надо определиться с темами проектируемых сайтов:

1) Салон красоты

2) Ветеринарная поликлиника

3) Адвокатский кабинет

4) Волонтерское движение

5) Ремонтная мастерская

6) Турфирма

7) Кондитерская по выпечке тортов на заказ

8) Агентство по продаже билетов

9) Приемная комиссия в учебное заведение

10) Кадровое агентство

11) Офис семейного врача

12) Охранное предприятие

13) Приемная депутата

14) Агентство по организации праздников

15) Свободная тема

**Практическая работа № 1**

Рис. 1 — Панель управления OpenServer

Знакомство с Open Server. Настройка домена.

Создание статичной страницы

Open Server Panel — программный комплекс имеет богатый набор серверного программного обеспечения, удобный, многофункциональный продуманный интерфейс, обладает мощными возможностями по администрированию и настройке компонентов. Платформа широко используется с целью разработки, отладки и тестирования веб-проектов, а так же для предоставления веб-сервисов в локальных сетях.

Сервис работает под управлением операционной системы windows, дистрибьютив можно скачать по адресу:

<https://ospanel.io/download/>

Полученный архив необходимо распаковать на локальный компьютер или переносной диск. C usb накопителей сервис может работать очень медленно или не работать вообще. Для виртуальной трансляции сайта необходимо создать тестовую страницу:

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Моя тестовая страница </title>

<!--<link rel="stylesheet" href="master.css">-->

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

</head>

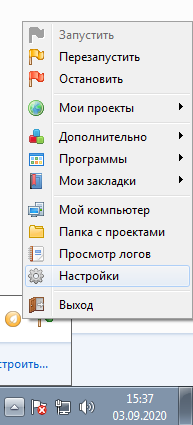
<body>

<h3>Ура все работает!!!</h3>

<p>Привет, ГБПОУ ПКГХ!</p>

</body>

</html>

В распакованном архиве создайте папку ~OpenServer/domains/testsite и сохраните туда тестовую страницу с именем index.html. Заглавная страница, которая должна транслироваться при наборе в браузере доменного имени должна иметь имя index. Запустите сервис из распакованного архива. Управление происходит в левой части панели управления. Если сервис исправен и работает там появляется зеленый флажок. Выберите «Настройка» - рис 1. Перейдите во вкладку — Домены. Произведите настройку как показано на рис. 2. и нажмите кнопку «Добавить».

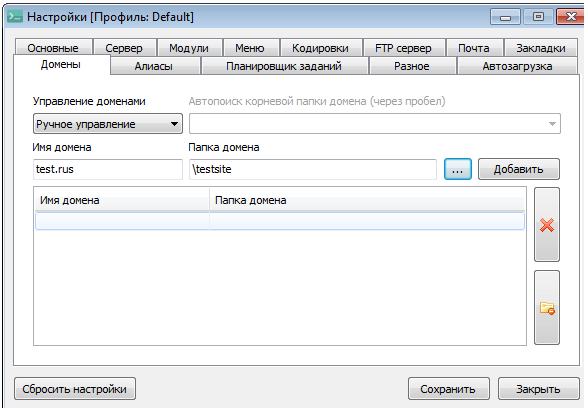


Рис. 2 – Настройка Домена

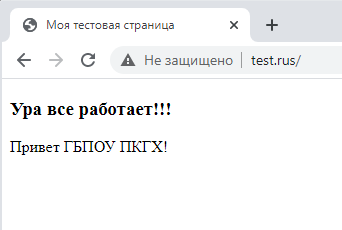
Откройте браузер и наберите в адресной строке заданное доменное имя — test.rus. Сайт транслируется — рис. 3.

Рисунок 3 — Трансляция сайта

**Цель работы:** Знакомство с сервисом OpenServer

Задание:

1) В папке domains создайте папку, где будут находиться файлы вашего сайта.

2) Придумайте своему сайту доменное имя и настройте OpenServer на трансляцию сайта

3) Создайте заглавную страницу

Рекомендуемая структура показана на рис. 4.

Поместите сюда название компании. Также можете добавить миссию компании

Можете добавить сюда информацию, почему нам надо авторизоваться

АВТОРИЗАЦИЯ

Логин:

Пароль:

Поместите сюда контактную информацию, привяжите ее к нижней части экрана

Оставаться на сайте

Зарегистрироваться

Забыли пароль

Войти

Рисунок 4 — Структура заглавной страницы

4) Проверьте страницу в браузере, используя OpenServer

**Вопросы для самостоятельной работы**

1) Ознакомьтесь и установите сервис WAMP для windows.

<https://www.wampserver.com/ru/>

Перед установкой надо установить компонент Visual Studio 2012 : VC 11 vcredist\_x64/86.exe

<http://www.microsoft.com/en-us/download/details.aspx?id=30679>

2) Ознакомьтесь и установите сервис LAMP для Linux Ubuntu

<https://losst.ru/ustanovka-lamp-ubuntu-18-04>

Не забудьте открыть порт 80 и разрешить пользователю, от имени которого запущен сервис, читать исполняемые файлы.

**Практическая работа № 2**

Отладка php кода. Настройка xdebug в NetBeans.IDE

**Теоретическая часть.** PHP является серверным языком программирования и отладка его скриптов непосредственно в браузере невозможна. Для этой цели мы будем использовать расширение xdebug.

Основной целью расширения является максимально возможное упрощение отладки PHP-скриптов и добавление в разработку на PHP таких удобств, как точки останова, пошаговое выполнение и наблюдение за выражениями. Настройку и отладку мы будем производить на примере NetBeans.IDE которую необходимо скачать и установить:

<https://netbeans.apache.org/download/index.html>

При установке OpenServer расширение xdebuge установлено, но не активировано. Для его активации следует открыть в панели управления OpenServer (рис. 1) файл php.ini (вкладка Дополнительно/Конфигурация/php7.x)

В этом файле раскомментировать (убрать вначале строки «;») и исправить следующие строки:

zend\_extension=/path/to/xdebug.so ;Это путь к расширению xdebug, он должен быть актуальным!

xdebug.remote\_enable=1

xdebug.remote\_handler=dbgp

xdebug.remote\_mode=req

xdebug.remote\_host=127.0.0.1

xdebug.remote\_port=9000

Сохраните файл и проверьте настройки. Перезапустите OpenSever (рис. 1) В панели управления выберите «Дополнительно/PHP информация». В окне браузера появятся все настройки PHP, в том числе и для xdebug (рис. 5). Проверьте их.



Рис. 5 — Настройки расширения xdebug

Затем необходимо настроить отслеживание действий пользователя в браузере. Из Интернет магазина Chrome или Mozilla нужно установить расширение Xdebug helper. Расширение активируется на отладку в правой части адресной строки браузера. Настройте расширение на работу с IDE NetBeans — рис. 6

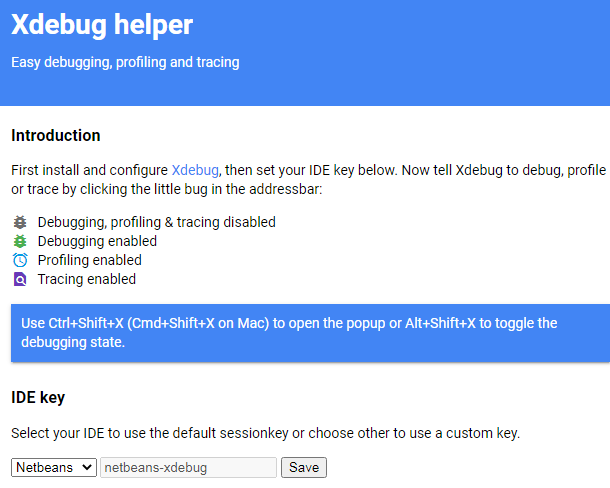


Рис. 6 — Настройка Xdebug helper

Завершающим этапом является настройка IDE NetBeams. Войдите в меню «Tools/Options». В окне настроек перейдите в PHP/Debugging и перенесите настройки из php.ini — рис. 7.

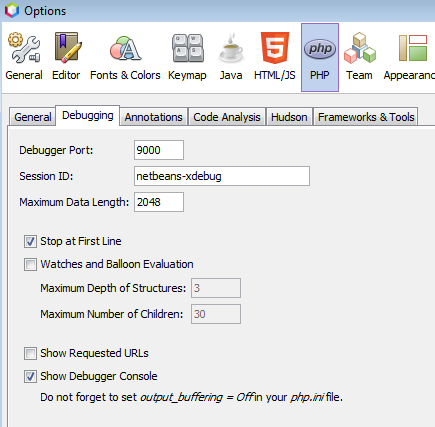


Рис. 7 — Настройка IDE NetBeans

При запуске отладки можно устанавливать точки остановки и просматривать значения переменных на разных этапах выполнения скрипта — рис. 8. Если отладчик не запустился попробуйте сменить порт, например, на 9010 в IDE NetBeans и php.ini. Порт 9000 может быть заблокирован, если используется другими программами.

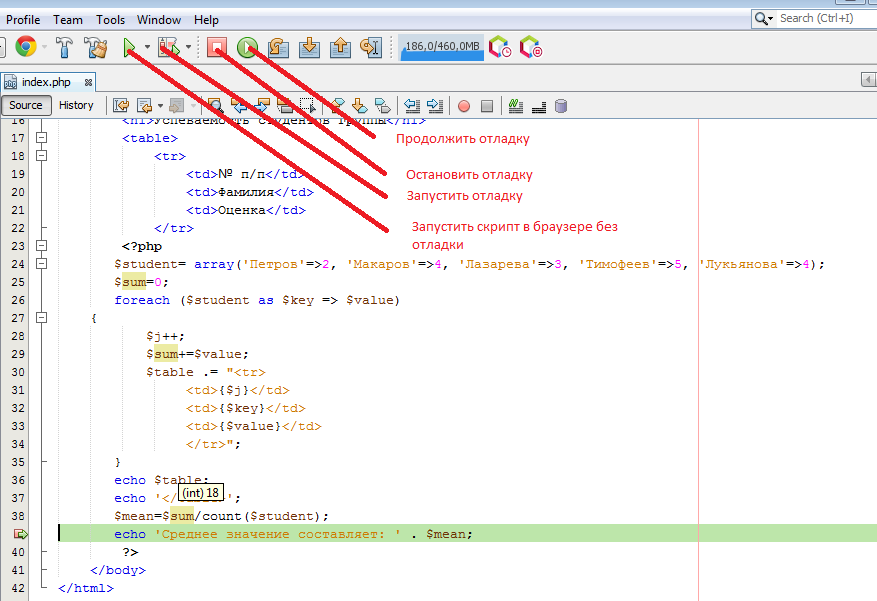


Рис. 8 — Отладка php скрипта

**Цель работы:** Освоить навыки отладки php кода

**Задание:**

1) Настройте xdebug для отладки php кода;

2) Изучите основные функции для работы с массивами:

<https://www.php.net/manual/ru/ref.array.php>

3) Создайте новый php скрипт и произведите его отладку

**1 Вариант**. Создайте ассоциативный массив содержащий данные о движении поездов на участке «Санкт-Петербург — Москва»:

Название поезда Время в пути, ч

Сапсан 4,0

№119А 9,3

Невский экспресс 4,2

Аврора 4,5

Юность 7,8

Красная стрела 8,0

Смена 8,2

После исследования пассажиропотоков администрация ФПК решила вывести из расписания поезда «Аврора» и «Юность». Выведите на страницу старые и новые сведения о движении поездов. Новые сведения нужно рассортировать по времени в пути.

**2 Вариант**. В магазине «Пятерочка» работает 5 кассиров. Сформируйте ассоциативный массив, содержащий сведения о их месячной выручке:

Имя кассира Месячная выручка, млн. руб

Валентина 0,92

Тамара 1,02

Александра 1,6

Антон 1,3

Мария 0,65

По результатам месячной работы им начисляется премия в размере 0,5% от суммы превышения выручку в 1 млн. руб. Выведите на веб странице сведения о кассирах, их выручке и размере премии.

**3 Вариант.** В начале обучения в колледже было сформировано две группы. Однако, в процессе обучения численность групп уменьшилась. Сформируйте ассоциативные массивы, содержащие ФИО студента и год рождения ИП-86-3:

1) Александров В.П 2070

2) Быков. П.П 2071

3) Волошин К.Ю. 2069

4) Прокопьев Т.В. 2060

5) Антонова С.А. 2068

ИП-86-4:

1) Тарасов Р.В. 2070

2) Иванова И.Р. 2070

3) Степин Р.О. 2068

4) Антонов Р.А 2069

5) Волкодав В.В. 2072

Администрация колледжа решила объединить эти две группы. Введите вместе с датами рождения гр. ИП-86-3, ИП-86-4, а также объединенную группу, которая получилась присоединением гр ИП-86-4 к группе ИП-86-3, в новую группу решил также перейти еще Ползунов И.С. (2060 г.р.) из другого колледжа.

**4 Вариант.** В библиотеку обращались студенты за учебниками, и им было выдано:

ФИО студента Кол. выданных учебников

Ларьков А.Л. 6

Павлов А.Д. 0

Алксандров А.Ж. 4

Иванова И.П. 3

Марков П.Н. 9

Сформируйте ассоциативный массив из этих данных. Эти учебники студенты оставили дома. Однако, преподаватель попросил сегодня дополнительно сегодня взять еще 5 учебников. Выведите сведения о количестве учебников у каждого студента до сегодняшнего дня и в конце сегодняшнего, с учетом того, что студенты старались выполнить задание, хотя бы частично, но библиотека не выдает больше 10 учебников.

**5 Вариант.** В магазине представлены следующие виды молока:

Марка Емкость, л

Пискаревское 0,9

Луговое 1,0

Петмол 0,85

Брест Литовское 1,0

Савушкин хуторок 0,94

Покупатель Валерий Сергеевич ошибочно полагает, что все молоко расфасовано в литровые пакеты. Сформируйте ассоциативный массив из ассортимента магазина, выведите его на страницу вместе с возможной ошибкой Валерия Сергеевича, с учетом того, что емкость пакета «Пискаревского» молока он внезапно вспомнил.

**Пример выполнения:** В группе писали практическую работу:

Фамилия Результат

Петров 2

Макаров 4

Лазарева 3

Тимофеев 5

Лукъянова 4

Сформируйте ассоциативный массив из полученных данных, выведите его на экран, а также определите среднюю успеваемость группы.

**Программа:**

<html>

<head>

<meta charset="UTF-8">

<title>Ведомость</title>

</head>

<body>

<style type="text/css">

table{

border-collapse: collapse;

}

td{

width: 20%;

border: 1px solid black;

}

</style>

<h1>Успеваемость студентов группы</h1>

<table>

<tr>

<td>№ п/п</td>

<td>Фамилия</td>

<td>Оценка</td>

</tr> <!--Это шапка таблицы-->

<?php

$student = array('Петров'=>2, 'Макаров'=>4, 'Лазарева'=>3, 'Тимофеев'=>5, 'Лукьянова'=>4);

$sum=0;

foreach ($student as $key => $value) // Переберем созданный ранее массив $student

{

$j++;

$sum+=$value;

$table .= "<tr>

<td>{$j}</td>

<td>{$key}</td>

<td>{$value}</td>

</tr>";

}

echo $table; //Вывод переменной, содержащей заполнение таблицы на страницу

echo '</table>';

$mean=$sum/count($student);

echo 'Среднее значение составляет: ' . $mean; //точка объединяет символьные переменные

?>

</body>

</html>

Результат работы программы — рис. 9

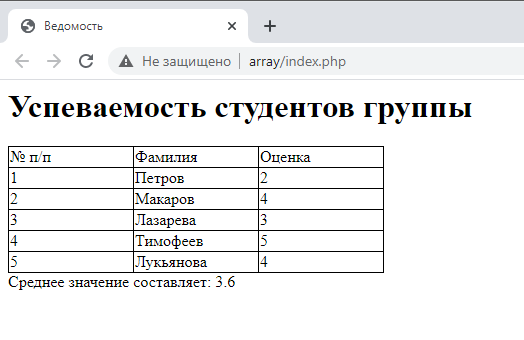


Рисунок 9 — Пример выполнения задания

**Вопросы для самостоятельной работы**

1) Настройте отладку php кода в IDE Atom

2) Настройте отладку php кода в PHP Storm

**Практическая работа № 3**

Настройка phpmyadmin. Работа с Designer, отображение связей таблиц, каскадное обновление данных таблиц

**Теоретическая часть.** Во вкладке *OpenServer* - «Настройки» выберите «Модули» и установите последнюю версию *mysql*. Затем во вкладке «Дополнительно» выберите *phpmyadmin*. Введите имя пользователя и пароль *root*. В верхней части страницы выберите «Учетные записи пользователей» и создайте нового пользователя со своим паролем: имя хоста — *localhost*; глобальные привилегии — отметить все, кроме *Grant*. Этот пользователь будет временно работать с базой данных.

Чтобы база данных была реляционной необходимо настроить связи. Для отображения связей служит расширение *Designer*. По умолчанию связи отключены. Для настройки следует изменить конфигурационный файл *phpmyadmin* (~\OpenServer\userdata\config\phpmyadmin\_config.inc.php), добавив в конце файла:

/\*\*

\* Servers configuration

\*/

$i = 0;

/\*\*

\* First server

\*/

$i++;

/\* Authentication type \*/

$cfg['Servers'][$i]['auth\_type'] = 'cookie';

/\* Server parameters \*/

$cfg['Servers'][$i]['host'] = 'localhost';

$cfg['Servers'][$i]['compress'] = false;

$cfg['Servers'][$i]['AllowNoPassword'] = false;

/\* Storage database and tables \*/

$cfg['Servers'][$i]['pmadb'] = 'phpmyadmin';

$cfg['Servers'][$i]['bookmarktable'] = 'pma\_\_bookmark';

$cfg['Servers'][$i]['relation'] = 'pma\_\_relation';

$cfg['Servers'][$i]['table\_info'] = 'pma\_\_table\_info';

$cfg['Servers'][$i]['table\_coords'] = 'pma\_\_table\_coords';

$cfg['Servers'][$i]['pdf\_pages'] = 'pma\_\_pdf\_pages';

$cfg['Servers'][$i]['column\_info'] = 'pma\_\_column\_info';

$cfg['Servers'][$i]['history'] = 'pma\_\_history';

$cfg['Servers'][$i]['table\_uiprefs'] = 'pma\_\_table\_uiprefs';

$cfg['Servers'][$i]['tracking'] = 'pma\_\_tracking';

$cfg['Servers'][$i]['userconfig'] = 'pma\_\_userconfig';

$cfg['Servers'][$i]['recent'] = 'pma\_\_recent';

$cfg['Servers'][$i]['favorite'] = 'pma\_\_favorite';

$cfg['Servers'][$i]['users'] = 'pma\_\_users';

$cfg['Servers'][$i]['usergroups'] = 'pma\_\_usergroups';

$cfg['Servers'][$i]['navigationhiding'] = 'pma\_\_navigationhiding';

$cfg['Servers'][$i]['savedsearches'] = 'pma\_\_savedsearches';

$cfg['Servers'][$i]['central\_columns'] = 'pma\_\_central\_columns';

$cfg['Servers'][$i]['designer\_settings'] = 'pma\_\_designer\_settings';

$cfg['Servers'][$i]['export\_templates'] = 'pma\_\_export\_templates';

/\*\*

\* End of servers configuration

\*/

Более подробную информацию о настройке phpmyadmin Вы можете узнать здесь:

~/OpenServer/modules/system/html/openserver/phpmyadmin/doc/html/config.html

Для работы Designer необходимо создать базу данных *phpmyadmin*. На заглавной странице *phpmyadmin* нажмите сверху «Импорт» и импортируйте файл: ~\OpenServer\modules\system\html\openserver\phpmyadmin\sql\create\_tables.sql

Создадим тестовую базу данных *social\_network* c двумя таблицами: *user* и *photo\_gallery* – рис. 10.

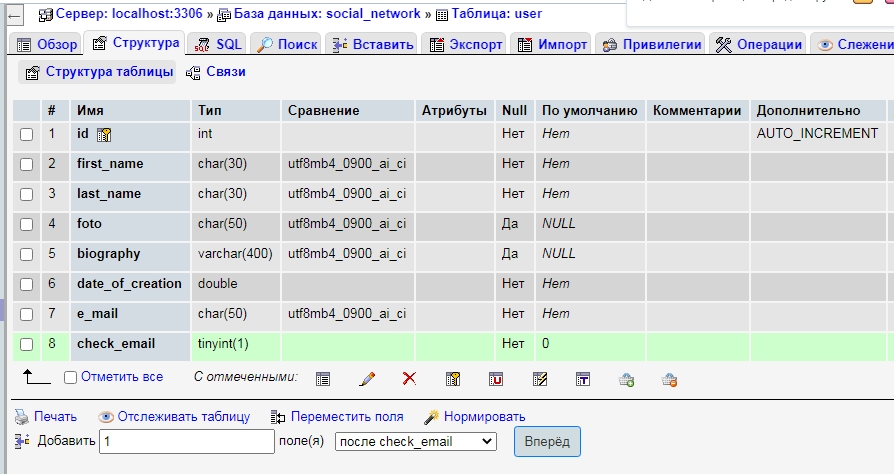


Рис. 10 – Таблица пользователи (user)

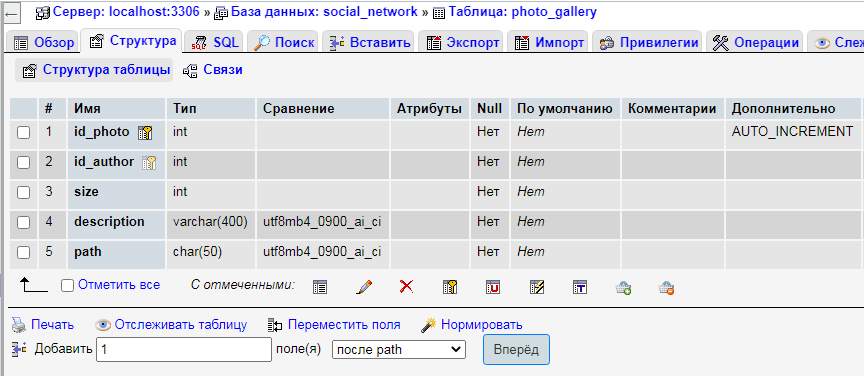


Рис. 11 – Таблица Фотогалерея (photo\_gallery)

К таблице *photo\_gallery* нужно добавить внешний ключ *id\_author*. Для этого откройте таблицу → Структура → Связи, см. рис. 12.

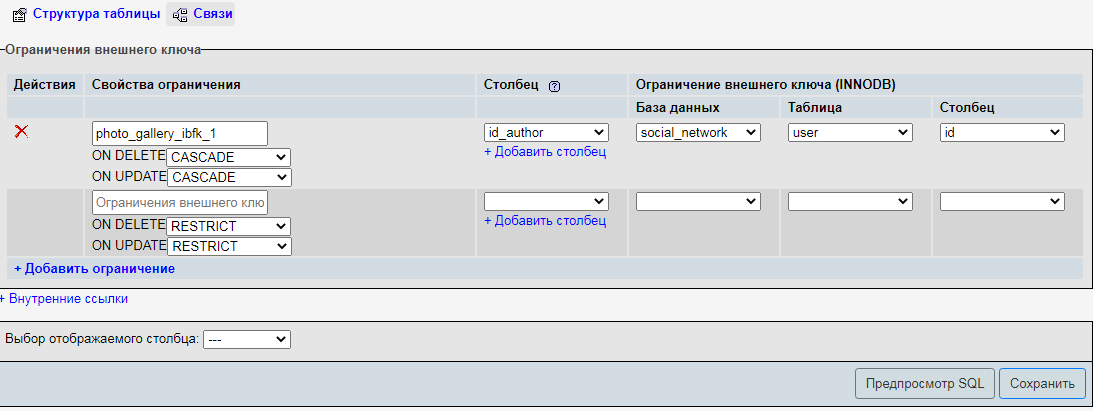


Рис. 12 – Добавление внешнего ключа

Выберите базу данных → Еще → Дизайнер. Структура базы данных показана на рис. 13. Теперь при удалении пользователя будут также удалены и связанные данные об изображениях.

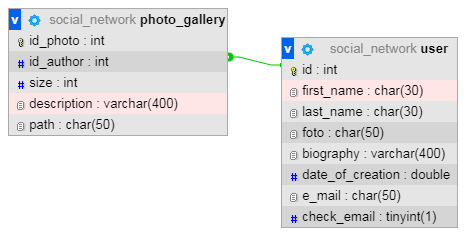


Рис. 13 – Структура базы данных

**Цель работы.** Освоить навыки администрирования СУБД *mysql* при помощи веб-приложения *phpmyadmin*.

**Задание.**

1. Настройте «Дизайнер» на отображение связей в таблицах;
2. Создайте таблицу, куда будут заноситься данные с формы регистрации. Кроме данных формы предусмотрите поля – время регистрации, проверка электронной почты, администратор (логическое);
3. Создайте еще 2…3 таблицы, отражающие функционал выбранной темы. В проектируемой базе данных должна содержаться 1…2 связи «Один-ко-многим» и 1…2 связи «Многие-ко-многим» ;
4. Проверьте результат при помощи «Дизайнер».

**Пример выполнения.**

Связь «Один-ко многим» реализована связью между таблицами «*user*» и «*photo\_gallery*». Связь «Многие-ко-многим» реализована с помощью двух связей «Один-ко-многим» и промежуточной таблицы – рис. 14.

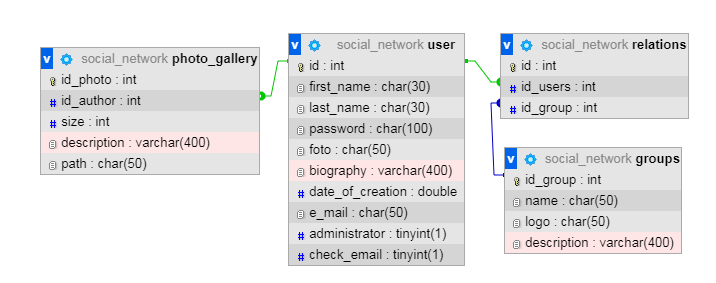


Рис. 14 – Пример выполнения задания

**Вопросы для самостоятельной работы**

1. Внесите нескольких строк в созданные таблицы
2. Проверьте работу связей между таблицами

**Практическая работа № 4**

Работа с данными формы, POST запросы

**Теоретическая часть.** Для отправки данных на сервер могут использоваться формы. Для отправки формы методом POST необходимо прописать ее атрибуты:

<form action="creat\_user.php" method="POST" enctype="multipart/form-data">

*action* – скрипт на сервере, обрабатывающий данные формы, *enctype* – указывает на характер передаваемых данных. Значение *multipart/form-data* позволяет передавать смешанное содержимое, в том числе файлы.

Данные поступающие на сервер хранятся в суперглобальной переменной *$\_POST*. Эта переменная является ассоциативным массивом, ключи и значения которого можно посмотреть при отладке кода во вкладке «Переменные» IDE NetBeans – рис. 15.

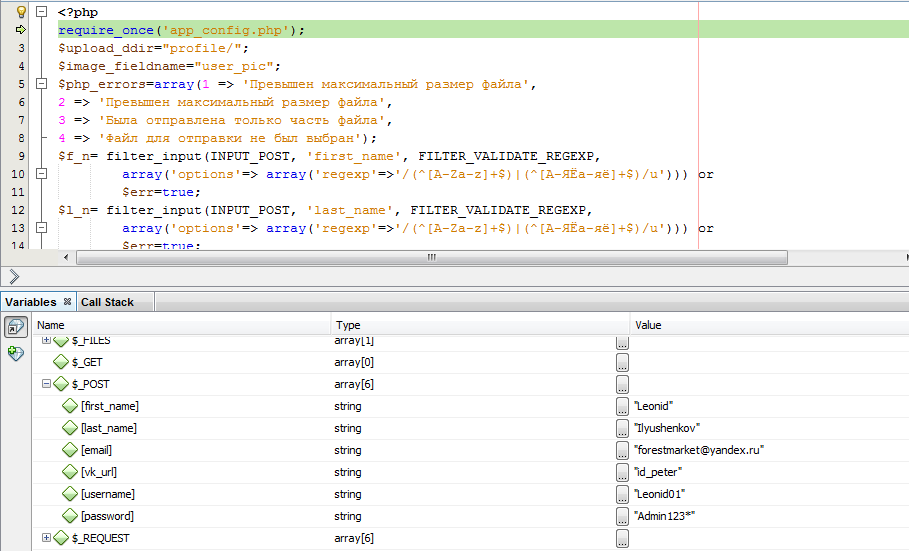


Рис. 15 – Значения суперглобальной переменной $\_POST

Чтение данных поступивших на сервер (рис.15)

<? php

$first\_name=$\_POST['first\_name'];

$last\_name=$\_POST['last\_name'];

$email=$\_POST['email'];

$vk\_url=$\_POST['vk\_url'];

$username=$\_POST['username'];

$password=$\_POST['password'];

?>

Указанные значения можно использовать для записи в базу данных и формирования страницы пользователя.

**Цель работы.** Научиться формировать и обрабатывать POST запросы

**Задание.**

1. В соответствии с темой создайте свою форму регистрации
2. В файле заглавной страницы (рис. 4) сделайте гиперссылку на страницу регистрации
3. Создайте php скрипт, получающий данные формы регистрации и формирующий страницу пользователя. Загрузку файлов мы произведем позднее. Необходимо предусмотреть поля формы для файлов и место на странице пользователя, куда будут помещены данные графических файлов

**Пример выполнения.** Форма регистрационных данных имеет табличную верстку из трех столбцов. Третий столбец используется для отметок валидации в следующих работах.

<html lang="ru">

<head>

<meta charset="utf-8" />

<title>Регистрация пользователя</title>

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

</head>

<body onload="controll()">

<header>Наша социальная сеть</header>

<h2>Введите свои данные</h2>

<fieldset>

<form action="show\_user.php" method="POST" enctype="multipart/form-data">

<table id="form">

<tr>

<td class="col1"><label for="first\_name">Имя:</label></td>

<td class="col1"><input type="text" name="first\_name"></td>

<td class="col2"></td></tr>

<tr>

<td></td><td class="hint">Введите имя латинскими или русскими буквами, минимум

3 буквы, максимум 20 букв</td>

</tr>

<tr>

<td><label for="last\_name">Фамилия:</td>

<td><input type="text" name="last\_name"></td>

<td></td></tr>

<tr>

<td></td><td class="hint">Введите фамилию латинскими или русскими буквами, минимум

3 буквы, максимум 20 букв</td>

</tr>

<tr>

<td><label for="user\_pic">Изображение профиля:</label></td>

<td><input type="file" name="user\_pic"></td>

<td></td></tr>

<tr>

<td></td><td class="hint">Выберите изображение в формате jpeg или bmp

размером не более 2 Мб</td>

</tr>

<tr>

<td><label for="email">e-mail:</label></td>

<td><input type="email" name="email"></td>

<td></td></tr>

<tr>

<td><label for="vk\_url">Адресс Вконтакте:</label></td>

<td><input type="text" name="vk\_url"></td>

<td></td> </tr>

<tr>

<td><label for="username">Имя пользователя:</label></td>

<td><input type="text" name="username"></td>

<td></td></tr>

<tr>

<td></td><td class="hint">От 3 до 20 символов: цифр, английских букв и знака \_</td>

</tr>

<tr>

<td><label for="password">Пароль:</label></td>

<td><input type="password" name="password"></td>

<td></td></tr>

<tr>

<td></td><td class="hint">От 6 до 20 символов, обязательно одна заглавная и строчная английская буква,

одна цифра и один спец. символ. Русские буквы не допускаются</td>

</tr>

<tr>

<td>Повторите пароль:</td>

<td><input type="password" size="20"> </td>

<td></td></tr>

</table>

</fieldset>

<br>

<table>

<tr>

<td class="button" ><input type="submit" value="Зарегистрироваться"

id="submit" class="button"></td>

<td class="button"><input type="reset" value="Очитстить" class="button"> </td>

</tr>

</table>

</form>

<br><br>

<footer>Ильюшенков Леонид Владимирович</footer>

</body>

</html>

Результат работы скрипта показан на рис. 16

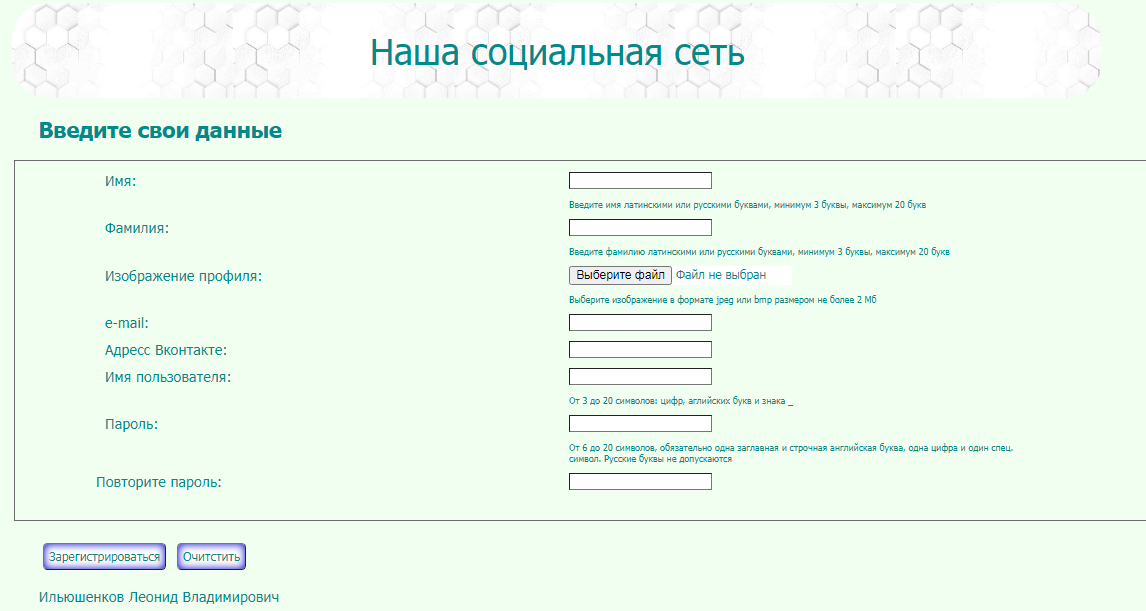


Рис. 16 – Форма регистрации

Данные этой формы читаются и обрабатываются скриптом:

<html lang="ru">

<head>

<meta charset="utf-8" />

<title>Данные пользователя</title>

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

<!---<script type="text/javascript"src="main1.js"></script>-->

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

</head>

<body>

<header>Наша социальная сеть</header>

<?php

$first\_name=$\_POST['first\_name'];

$last\_name=$\_POST['last\_name'];

$email=$\_POST['email'];

$vk\_url=$\_POST['vk\_url'];

$username=$\_POST['username'];

$password=$\_POST['password'];

?>

<table class='head'>

<tr><td rowspan="2">Здесь будет ваше изображение</td>

<td></td></tr>

<tr><td><p>Добро пожаловать, <strong><?php echo $first\_name . ' ' . $last\_name ?></strong></p>

<?php echo '<a href=unautoriz.php><strong>Выйти из приложения</strong></a>';?>

<p>Ваши регистрационные данные:</p><br>

<?php $check=0;

if ($check!=1){

echo "<p class='error'>Адрес электронной почты не подтвержден, зайдите на почту. Аккаунт будет удален

в течении 30 дней, в случае отсутствия подтверждения!</p>";

}?>

</td></tr>

</table>

<br>

<table class='New'>

<tr><td>ID</td><td id='id\_u'></td></tr>

<tr><td>Логин</td><td><?php echo $username?></td></tr>

<tr><td>Имя</td><td><?php echo $first\_name ?></td></tr>

<tr><td>Фамилия</td><td><?php echo $last\_name ?></td></tr>

<tr><td>Электронная почта</td><td><?php echo $email ?></td></tr>

<tr><td>Адрес Вконтаке</td><td><?php echo '<a href=' . $vk\_url . '>' . $vk\_url . '</a>' ?></td></tr>

</table>

<br>

</body>

</html>

Пример страницы пользователя показан на рис. 17



Рис. 17 – Регистрационные данные пользователя

**Практическая работа № 5**

Регулярные выражения

**Теоретическая часть.** Оставлять форму в том виде, который приводится в практической работе № 4, нельзя. Нужно помочь пользователю правильно заполнить форму и по возможности уберечь базу данных от хранения неадекватных данных. Для этих целей будем использовать регулярные выражения.

В *php* регулярные выражения помещаю между //. В шаблонах используются следующие метасимволы (символы обозначающие группы других символов) часто называемые *egrep* - стандартом:

\ - считать следующий метасимвол как обычный символ.

^ - начало строки

. - один произвольный символ. Кроме '\n' - конец строки.

$ - конец строки

| - альтернатива (или)

() - группировка

[] - класс символов

Метасимволы имеют модификаторы (пишутся после метасимвола):

\* - повторяется 0 или большее число раз

+ - -//- 1 или большее число раз

? - 1 или 0 раз

{n} - точно n раз

{n,} - по меньшей мере раз

{n,m} - не меньше n, но и не больше m

Во все других случаях фигурные скобки считаются обычными (регулярными) символами. Таким образом '\*' эквивалентна {0,} , '+' - {1,} и '?' - {0,1}. n и m не могут быть больше 65536.

Шаблоны работают так же, как и двойные кавычки, поэтому в них можно использовать `\` - символы (бэкслэш-символы):

\t - символ табуляции

\n - новая строка

\r - перевод каретки

\а - перевод формата

\v - вертикальная табуляция

\e - escape

Дополнительно в Perl добавлены следующие метасимволы:

\w - алфавитно-цифровой или '\_' символ

\W - не -//-

\s - один пробел

\S - один не пробел

\d - одна цифр

\D - одна не цифра

Обратите внимание, что все это «один» символ. Для обозначения последовательности применяйте модификаторы. Так:

\w+ - слово

\d+ - целое число

[+-]?\d+ - целое со знаком

[+-]?\d+\.?\d\* - число с точкой

Кроме того существуют мнимые метасимволы. Обозначающие не существующие символы в месте смены значения. Такие как:

\b - граница слова

\B - не граница слова

\A - начало строки

\Z - конец строки

\G - конец действия m//g

Граница слова (\b) - это мнимая точка между символами \w и \W. Внутри класса символов '\b' обозначает символ backspace (стирания). Метасимволы \A и \Z - аналогичны '^' и '$', но если начало строки '^' и конец строки '$' действуют для каждой строки в многострочной строке, то \A и \Z обозначают начало и конец всей многострочной строки.

Модификаторы ставятся после //

i Регистронезависимость. Например "/a/i" ищет и a, и A.

u Использование кодировки utf-8 (для поиска русского текста например).

m Многострочный поиск.

Perl версии 5 содержит дополнительные конструкции шаблонов:

(?#комментарий) - комментарий в теле шаблона.

(?:шаблон) - группировка как и '( )', но без обратной ссылки

(?=шаблон) - "заглядывание" вперед. Например /\w+(?=\t)/ соответствует слову, за которым идет табуляция, но символ '\t' не включается в результат.

Для проверки корректности работы регулярных выражений можно использовать сайт:

<https://uvsoftium.ru/php/regexp.php>

Примеры регулярных выражений:

|  |  |
| --- | --- |
| /^[A-Za-z]+$)|(^[А-ЯЁа-яё]+$/ u | Выражение должно содержать только английские или русские буквы и букву ё. Буквы могут быть заглавными или строчными |
| /^(vk.com).{3,}$/ | В начале строки vk.com и еще не менее трех символов в конце |
| /^[\w]+$/ | Выражение может содержать буквы, цифры и знак \_ |
| /^(?=.\*[A-Z])(?=.\*[\d])(?=.\*[a-z])(?=.\*[\W])[a-zA-Z\d\W]{6,20}$/ | Выражение должно содержать одну или несколько заглавных латинских буква, одну или несколько строчных латинских букв, одну или несколько цифр, один или несколько специальных символов. Длина выражения 6…20 символов |

Другие примеры можете посмотреть тут:

<http://archive-ipq-co.narod.ru/l1/regexp.html>

**Цель работы**: Освоить валидацию данных при помощи html5, javascript и php.

**Задание.**

1. Настройте валидацию созданной ранее формы
2. Создайте процедуры JavaScript помогающие пользователю заполнить формы

**Пример выполнения**

Валидация полей формы произведена при помощи атрибутов тега input. Перечень и назначение атрибутов можно посмотреть здесь:

<http://htmlbook.ru/html/input>

Программный код имеет вид:

<html lang="ru">

<head>

<meta charset="utf-8" >

<title>Регистрация</title>

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

<link rel="stylesheet" href="fonts/font-awesome-4.7.0/css/font-awesome.css">

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

</head>

<body onload="controll()">

<header>Наша социальная сеть</header>

<h2>Введите свои данные</h2>

<fieldset>

<form action="create\_user.php" method="POST" enctype="multipart/form-data">

<table id="form">

<tr>

<td class="col1"><label for="first\_name">Имя:</label></td>

<td class="col1"><input type="text" name="first\_name" minlength="3" maxlength="20" required

pattern="(^[A-Za-z]+$)|(^[А-ЯЁа-яё]+$)" onblur="validate(0)"></td>

<td class="col2"></td></tr>

<tr>

<td></td><td class="hint">Введите имя латинскими или русскими буквами, минимум

3 буквы, максимум 20 букв</td>

</tr>

<tr>

<td><label for="last\_name">Фамилия:</td>

<td><input type="text" name="last\_name" required pattern="(^[A-Za-z]+$)|(^[А-ЯЁа-яё]+$)"

minlength="3" maxlength="20" onblur="validate(2)"></td>

<td></td></tr>

<tr>

<td></td><td class="hint">Введите фамилию латинскими или русскими буквами, минимум

3 буквы, максимум 20 букв</td>

</tr>

<tr>

<td><label for="user\_pic">Изображение профиля:</label></td>

<td><input type="file" name="user\_pic" size="50" onchange="filecontrol(4)"

required id="file"></td>

<td></td></tr>

<tr>

<td></td><td class="hint">Выберите изображение в формате jpeg или bmp

размером не более 2 Мб</td>

</tr>

<tr>

<td><label for="email">e-mail:</label></td>

<td><input type="email" name="email" onblur="validate(6)" required></td>

<td></td></tr>

<tr>

<td><label for="vk\_url">Адресс Вконтакте:</label></td>

<td><input type="text" name="vk\_url" required pattern="^(vk.com).{3,}$" minlength="3"

maxlength="20" onblur="validate(7)"></td>

<td></td> </tr>

<tr><td></td><td class="hint">Начните адрес с vk.com/</td><td></td></tr>

<tr>

<td><label for="username">Имя пользователя:</label></td>

<td><input type="text" name="username" minlength="3" maxlength="20" required

pattern="^[\w]+$" onblur="uniq\_username(this.value, this.validity.valid)"></td>

<td></td></tr>

<tr>

<td></td><td class="hint">От 3 до 20 символов: цифр, аглийских букв и знака \_</td>

</tr>

<tr>

<td><label for="pasword">Пароль:</label></td>

<td><input type="password" name="password" minlength="6" maxlength="20" required onblur="validate(11)"

pattern="^(?=.\*[A-Z])(?=.\*[\d])(?=.\*[a-z])(?=.\*[\W])[a-zA-Z\d\W]{6,20}$"</td>

<td></td></tr>

<tr>

<td></td><td class="hint">От 6 до 20 символов, обязательно одна заглавная и строчная английская буква,

одна цифра и один спец. символ. Русские буквы не допускаются</td>

</tr>

<tr>

<td>Повторите пароль:</td>

<td><input type="password" size="20" id="pass2" onblur="checkpass(13)" required </td>

<td></td></tr>

</table>

</fieldset>

<br>

<table>

<tr>

<td class="button" ><input type="submit" value="Зарегистрироваться" disabled

id="submit" class="button"></td>

<td class="button"><input type="reset" value="Очитстить" class="button"> </td>

</tr>

</table>

</form>

<br><br>

<footer>Ильюшенков Леонид Владимирович</footer>

</body>

</html>

При работе скрипта загружается форма рис. 18. В третьем столбце таблицы формы помещены метки валидации. Они взяты из иконочного шрифта Font Awesome. Который можно скачать и положить в папку проекта:

<https://fontawesome.ru/get-started/>

В рассматриваем коде подключен css код шрифта. Здесь использовались символы: *check* – верно и *times* – ошибка.

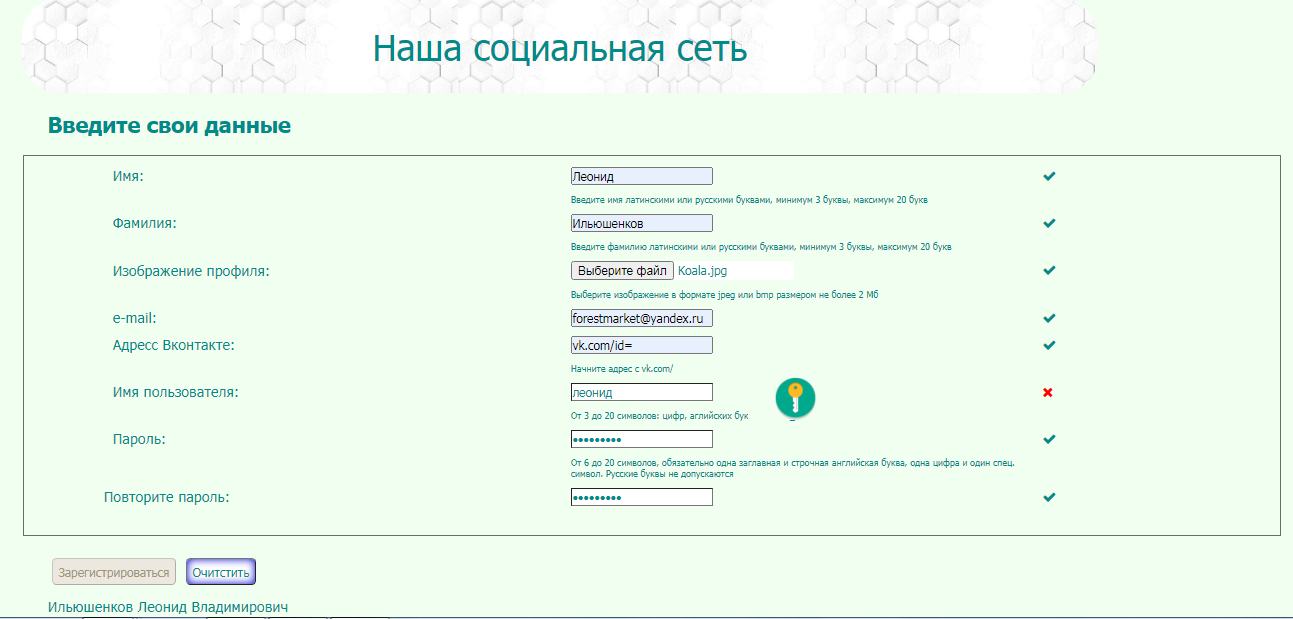


Рис. 18 – Страница регистрации с валидацией данных

В тег, где должны отобразиться эти символы нужно прописать классы: *fa fa-check* или *fa fa-times*. Для корректного отображения символов в файле *font-awesome.css* был дополнительно прописан цвет элемента times, цвет другого символа наследуется от страницы:

.fa-times{

color: red;

}

При необходимости в этом файле можно прописать другие свойства. Для управления страницей были созданы функции java-script:

//Эта функция отслеживает значение свойство validity объекта input и в зависимости от результата (result) помещает в третий столбец (от 0 второй) строки row отметку валидации

function validate(row){

a=document.getElementById('form');

result=a.rows[row].cells[1].firstElementChild.validity.valid;

if (result) {

a.rows[row].cells[2].innerHTML='<span class="fa fa-check"></span>';

}

else{

a.rows[row].cells[2].innerHTML='<span class="fa fa-times"></span>';

}

controll();

if (row==11){checkpass(13);} //если проверяется пароль, запускается проверка «Повторить //пароль»

}

function uniq\_username(username, valid) {

// Сюда нужен код проверки существования пользователя с таким именем

if (valid){

a.rows[9].cells[2].innerHTML='<span class="fa fa-check"></span>';

}

else {

a.rows[9].cells[2].innerHTML='<span class="fa fa-times"></span>';

}

controll();

}

function filecontrol(row) { //Проверка файла на тип и размер

a=document.getElementById('form');

b=a.rows[row].cells[1].firstElementChild;

size=b.files[0].size;

type=b.files[0].type;

if ((type=='image/png' || type=='image/jpeg' || type=='image/jpg') && size<2097152) {

a.rows[row].cells[2].innerHTML='<span class="fa fa-check"></span>';

}

else{

a.rows[row].cells[2].innerHTML='<span class="fa fa-times"></span>';

}

controll();

}

function checkpass(row) { //Проверка совпадения пароля

a=document.getElementById('form');//таблица внутри формы

b=a.rows[row].cells[1].firstElementChild.value;

c=a.rows[row-2].cells[1].firstElementChild.value;

if (b==c){

a.rows[row].cells[2].innerHTML='<span class="fa fa-check"></span>';

}

else{

a.rows[row].cells[2].innerHTML='<span class="fa fa-times"></span>';

}

controll();

}

function controll() {//Подсчет значков положительной валидации, если их 8 активируется кнопка «Зарегистрироваться»

rows=[0, 2, 4, 6, 7, 9, 11, 13]; // в этих строках должны быть значки валидации

a=document.getElementById('form');

n=0;

rows.forEach(function(item, i, rows){

try {

clasz= a.rows[item].cells[2].firstChild.className;

if (clasz=="fa fa-check") {

n++;

}}

catch (err){}//Если произошла ошибка, нет класса тега (значок отсутствует), то программа ничего не делает и идет дальше

});

if (n==8) {

document.getElementById('submit').disabled='';

}

else{

document.getElementById('submit').disabled=true}

}

**Вопросы самостоятельной работы**

1. Найдите и изучите виды иконочных шрифтов, отличных от Font Awesome;
2. Изучите другие подходы к проверке валидности формы:

[https://developer.mozilla.org/ru/docs/Learn/HTML/Forms/%D0%92%D0%B0%D0%BB%D0%B8%D0%B4%D0%B0%D1%86%D0%B8%D1%8F\_%D1%84%D0%BE%D1%80%D0%BC%D1%8B](https://developer.mozilla.org/ru/docs/Learn/HTML/Forms/Валидация_формы)

**Практическая работа № 6**

Подключение и работа с базой данных mysql. GET запросы

Для работы с базой данных в php 7.x используют расширение *mysqli*. Расширение *mysql* считается устаревшим и его использовать не следует. Подключение к базе данных осуществляется следующим образом:

$connect=mysqli\_connect($host, $user, $password, $database);

где: $*host* – имя хоста, в нашем случае *localhost*; *$user* – имя пользователя; $*password* – пароль, имя пользователя и пароль были определены в практической работе № 3; *$database* – база данных. Аргументы функции являются символьными переменными, их значение берется в кавычки. Для работы с русскими буквами, задайте кодировку:

mysqli\_set\_charset($connect, "utf8");

Выполнение запроса осуществляется следующим образом:

$result=mysqli\_query($connect, $query);

где *$connect* – параметры соединения с базой данных; *$query* – текст запроса. Запросы формируются следующим образом:

* Выборка из таблицы:

$query=’SELECT data FROM table’;

В запросах служебные слова рекомендуют писать заглавными буквами, параметры – строчными. Выбрать данные *data* из таблицы *table*. Например:

$query=′SELECT first\_name, last\_name FROM users′;

Выбрать имена *first\_name* и фамилии *last\_name* из таблицы «пользователи» *users*. В этом случае в переменной *$result* содержится объект *SQ*L, для его преобразования в массив *php* используют функцию *mysqli\_fetch\_array($result)*. Сформированный массив перебирается в цикле *while*:

$query=′SELECT first\_name, last\_name FROM users′;

$result=mysqli\_query($connect, $query);

while ($row = mysqli\_fetch\_array($result)) {

$first\_name=$row['first\_name'];

$last\_name=$row['last\_name'];

echo 'Имя: ' . $first\_name . ', Фамилия:' . $lastname . \n;

}

* Вставка данных в таблицу:

$query=″INSERT INTO table(col\_name) VALUES (data)″;

Вставить в поля *col\_name* таблицы *table* значения *data*. Например:

$query=″INSERT INTO users(first\_name, last\_name) VALUES (′Петр′, ′Павлов′)″;

Вставить в поля *first\_name*, *last\_name* таблицы *users* значения «Петр» и «Павлов». В этом примере двойные кавычки экранируют одинарные. Пароли в базу данных вставляют хешированными, для этого используют функцию:

$password\_hash= password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

где *PASSWORD\_DEFAULT* – алгоритм необратимого шифрования. Для проверки пароля используют функцию:

password\_verify($password , $hash)

функция возвращает *true*, если *$hash* соответствует *$password*.

* Обновить данные в таблицы:

$query=′UPDATE table SET col\_name1 = value1, col\_name2 = value2, ... [WHERE condition]′;

Установить значения таблицы *table* в поле *col\_name1* – *value1*, в поле *col\_name2* – *value2* … при выполнении условия *condition*. Например:

$query=″UPDATE users SET check\_email=1 WHERE email=′forestmarket@yandex.ru′″;

Установить в таблице *users* значение в поле *check\_email* равным *1*, везде, где поле *email* принимает значение ′*forestmarket@yandex.ru*′.

* Удалить значения:

DELETE FROM users WHERE id=2;

Удалить из таблицы *users* запись в которой *id*=2.

Для того чтобы узнать, под каким ключом (*id*) сделана новая запись, воспользуйтесь функцией:

$id=mysqli\_insert\_id($connect);

Если при выполнении запроса произошла ошибка, ее описание можно получить с помощью функции:

mysqli\_error($сonnect);

После окончания работы с базой данных, соединение нужно закрыть:

mysqli\_close($connect);

При составлении запросов, их следует предварительно тестировать в консольной версии mysql — выберите «Дополнительно/консоль», рис. 19.

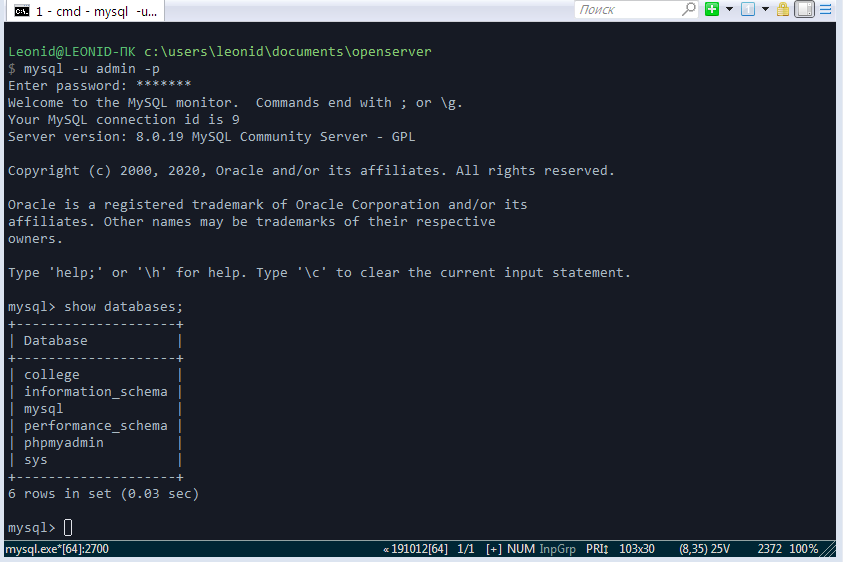


Рис. 19 — Консоль для тестирования *sql* запросов

Данные между страницами можно передавать через URL — GET запросы. Такая передача является менее защищенной по сравнению с POST запросами, так как URL адрес пользователь видит и может корректировать:

header("Location: mynetwork.rus/show\_photo.php?id=1034&photo=1");

Здесь задано перенаправление на сценарий *show\_photo.php*, которому передаются параметры *id*=1034 и *photo*=1. Чтение этих параметров производится из суперглобальной переменной *$\_GET*.

**Цель работы:** Освоить навыки составления запросов к базе данных *mysql*.

**Задание**:

1) Создайте сценарий, который вносит данные формы в базу данных;

2) После внесения данных, сделайте перенаправление на страницу пользователя, заполнение страницы следует осуществить чтением данных из базы;

3) Сделайте отображение пользователя при вводе логина и пароля.

**Пример выполнения:**

<?php

//чтение формы

$first\_name=$\_POST['first\_name'];

$last\_name=$\_POST['last\_name'];

$name=$f\_n . " " . $l\_n;

$email=$\_POST['email'];

$vk=$\_POST['vk\_url'];

$username=$\_POST['username'];

$position=strpos($vk, "https://");

if ($position===false){$vk="https://" . $vk;}

$password\_hash= password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

// запись в базу

$cn=mysqli\_connect('localhost', 'admin1', 'admin123', 'college');

mysqli\_set\_charset($cn, "utf8");

$inset\_student="INSERT INTO student(first\_name, last\_name, password, email, vk\_url, user\_name)

VALUES('{$first\_name}', '{$last\_name}', '{$password\_hash}', '{$email}', '{$vk}', '{$username}')" ;

$msqq=mysqli\_query($cn, $inset\_student)or

die(mysqli\_error($cn)) ;

$num=strval(mysqli\_insert\_id($cn));

//перенаправление

header("Location: show\_user.php?user\_id=" . $num); ?>

//перенаправление

header("Location: show\_user.php?user\_id=" . $num); ?>

*Страница пользователя*

<html lang="ru">

<head>

<meta charset="utf-8" />

<title>Моя страница</title>

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

</head>

<body>

<header>Моя социальная сеть</header>

<?php

$user\_id=$\_GET['user\_id'];

$select\_query="SELECT \* FROM student WHERE user\_id=" . $user\_id;//\* означает выбрать все

$cn=mysqli\_connect('localhost', 'admin1', 'frst123', 'college');

mysqli\_set\_charset($cn, "utf8");

$result=mysqli\_query($cn, $select\_query);

if ($result){

$row=mysqli\_fetch\_array($result);

$first\_name=$row['first\_name'];

$last\_name=$row['last\_name'];

$email=$row['email'];

$vk=$row['vk\_url']; }

?>

<br>

<table class='head'>

<tr><td rowspan="2">Здесь должно быть ваше изображение</td>

<td></td></tr>

<tr><td><p>Добро пожаловать, <strong><?php echo $first\_name . ' ' . $last\_name ?></strong></p>

<p>Выши регистрационные данные:</p><br>

<?php $check=0;

if ($check!=1){

echo "<p class='error'>Адрес электронной почты не подтвержден, зайдите на почту. Аккаунт будет удален втечении 30 дней, в случае отсутствия подтверждения!</p>";

}?>

</td></tr>

</table>

<br>

<table class='New'>

<tr><td>ID</td><td id='id\_u'><?php echo $user\_id ?></td></tr>

<tr><td>Имя</td><td><?php echo $first\_name ?></td></tr>

<tr><td>Фамилия</td><td><?php echo $last\_name ?></td></tr>

<tr><td>Электронная почта</td><td><?php echo $email ?></td></tr>

<tr><td>Адрес Вконтаке</td><td><?php echo '<a href=' . $vk . '>' . $vk . '</a>' ?></td></tr>

</table>

<br>

<a href=unautoriz.php><strong>Выйти из приложения</strong></a>

</body>

</html>

**Вопросы для самостоятельной работы**

1) Изучите другие возможности mysql: <http://www.mysql.ru/>

**Практическая работа № 7**

Безопасная загрузка данных форм,

технологии предотвращения *sql* инъекций

**Теоретическая часть.** Разработка web приложений должна вестись неразрывно с анализом возможных угроз. Так как формирование *sql* запроса производится в текстовой строке путем конкатенации служебных слов и данных формы, пользователь имеет возможность ввести в поле формы данные существенно изменяющие содержание запроса. Рассмотрим пример:

SELECT name, status, books FROM members WHERE name = 'Demo' AND password ='111'

Предположим, пользователь вместо имени «Demo» ввел «Demo' --», тогда выражение примет вид

SELECT name, status, books FROM members WHERE name = 'Demo' -- ' AND password ='111'

В этом выражении «--» означает комментарий и дальнейшую часть строки программа не учитывает. Таким образом, удалось обойти проверку пароля. Поэтому любые данные поступающие на сервер из вне, необходимо тщательным образом проверять. Для этого можно использовать функции фильтрации: *filter\_input* для фильтрации данных запросов и *filter\_var* для фильтрации переменных. Эти функции работают схожим образом:

filter\_input ($type, $variable\_name, FILTER)

где *$type* – константа, ссылающаяся на суперглобальную переменную, принимает значения: *INPUT\_GET*, *INPUT\_POST*, *INPUT\_COOKIE* и др; *$variable\_name* – имя получаемой переменной, *FILTER* – тип фильтра. Могут применяться следующие типы фильтров:

* *FILTER\_VALIDATE\_BOOLEAN* – пропускает только логические выражения;
* *FILTER\_VALIDATE\_EMAIL* – пропускает только адреса электронной почты;
* *FILTER\_VALIDATE\_FLOAT* – пропускает вещественные числа;
* *FILTER\_VALIDATE\_INT* – пропускает целые числа;
* *FILTER\_VALIDATE\_IP* – пропускает IP адреса;
* *FILTER\_VALIDATE\_REGEXP* – пропускает выражения, соответствующие регулярному выражению.

Примеры:

$email= filter\_input(INPUT\_POST, 'email', FILTER\_VALIDATE\_EMAIL);

$first\_name= filter\_input(INPUT\_POST, 'first\_name', FILTER\_VALIDATE\_REGEXP,

array('options'=> array('regexp'=>'/(^[A-Za-z]+$)|(^[А-ЯЁа-яё]+$)/u')));

Для экранирования специальных символов, которые могут исказить суть запроса можно использовать функцию *mysqli\_real\_escape\_string($connect, $variable)*. Здесь *$connect* – ссылка на соединение, *$variable* – переменная, значение, которой надо экранировать.

Эффективным средством борьбы с *sql* инъекциями являются подготовленные *sql* выражения, они создаются функцией:

$stmt=mysqli\_prepare ($connect , $query );

В выражении *$query* вместо значения параметра следует поставить «*(?)*»:

$stmt= mysqli\_prepare($connect, 'SELECT user\_id FROM student WHERE user\_name=(?)');

Для подстановки параметра используют функцию:

mysqli\_stmt\_bind\_param($stmt, 's', $user\_name);

где '*s*' – тип переменной (*string*); '*i*' – *integer*; '*d*' – *double*; '*b*' – двоичный объект *blob*.

Для запуска выражения *sql* используют:

$answ= mysqli\_stmt\_execute($stmt);

Привязка переменных к подготовленному запросу для размещения результата:

mysqli\_stmt\_bind\_result($stmt, $user\_id);

Связывает результаты подготовленного выражения с переменными:

mysqli\_stmt\_fetch($stmt);

Если при выполнении запроса появилась ошибка, ее описание здесь:

mysqli\_stmt\_error($stmt);

Эффективным средством борьбы с взломом – ограничение прав пользователя базы данных. Дайте пользователю базы данных только те права, которые ему необходимы для выполнения запросов программы. Так же можно создать двоих пользователей: один на добавления данных в базу, другого на чтение.

**Цель работы:** Освоить технологии защиты от *sql* инъекций.

**Задание.**

1. Организуйте фильтрацию данных формы;
2. Скорректируйте *sql* запросы на запросы с подготовленными выражениями
3. Ограничьте права пользователя базы данных до действий, необходимых для обработки запросов приложения.

**Пример выполнения:**

<?php

//Фильтрация данных формы

$first\_name= filter\_input(INPUT\_POST, 'first\_name', FILTER\_VALIDATE\_REGEXP,

array('options'=> array('regexp'=>'/(^[A-Za-z]+$)|(^[А-ЯЁа-яё]+$)/u')));

$last\_name= filter\_input(INPUT\_POST, 'last\_name', FILTER\_VALIDATE\_REGEXP,

array('options'=> array('regexp'=>'/(^[A-Za-z]+$)|(^[А-ЯЁа-яё]+$)/u')));

$username= filter\_input(INPUT\_POST, 'user\_name', FILTER\_VALIDATE\_REGEXP,

array('options'=>array('regexp'=>'/^[\w]+$/')));

$email= filter\_input(INPUT\_POST, 'email', FILTER\_VALIDATE\_EMAIL);

$vk= filter\_input(INPUT\_POST, 'vk\_url', FILTER\_VALIDATE\_REGEXP,

array('options'=> array('regexp'=>'/^(vk.com).{3,}$/')));

$position=strpos($vk, "https://");

if ($position===false){$vk="https://" . $vk;}

$password\_hash= password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

require 'connect.php'; // подключение к базе данных

$mysqli\_set\_charset = mysqli\_set\_charset($cn, "utf8");

$Date\_reg= time();

//Добавление пользователя при помощи подготовленного выражения

$stmt= mysqli\_prepare($cn, 'INSERT INTO student(first\_name, last\_name, email, vk\_url, user\_name, password, Date\_reg) VALUES((?), (?), (?), (?), (?), (?), (?))');

mysqli\_stmt\_bind\_param($stmt, 'ssssssi', $first\_name, $last\_name, $email, $vk, $username, $password\_hash, $Date\_reg);

$answ= mysqli\_stmt\_execute($stmt);

mysqli\_close($cn);

header("Location: show\_user.php"); ?>

**Вопросы для самостоятельной работы**

1. Изучите эффективные способы борьбы с *sql* инъекциями при помощи расширения *PDO*.

**Практическая работа № 8**

Обработка ошибок работы приложения

**Теоретическая часть.** При работе приложения могут возникать самые разные ошибки: не удалось подключиться к базе данных, не получен ответ сервера, ошибка во входных данных и пр. Необходимо создать страницу, корректно объясняющую пользователю суть проблемы. Для перенаправления пользователя на функцию при возникновении ошибки используют оператор *or*:

$cn=mysqli\_connect('localhost', 'admin', 'Admin123\*', 'social\_network')

or handl\_error("Возникла ошибка при подключении к базе данных", mysqli\_connect\_error($cn));

При невозможности подключения к базе данных «Социальная сеть» идет перенаправление на пользовательскую функцию обработки ошибок *handl\_error*. Аргументами функции являются: сообщение пользователю и техническое описание проблемы. Техническое описание необходимо в процессе отладки приложения. В работающем приложении его следует скрыть, так как его содержание может быть полезно злоумышленникам.

Бывают случаи, когда нужно выполнить действие, игнорируя ошибку. В этом случае перед функцией ставят оператор - @:

$ans= @mysqli\_query($connect, $query);

При ошибке запроса не произойдет остановка приложения, вывод ошибок или предупреждений.

**Цель работы.** Научиться обрабатывать ошибки приложения.

**Задание:**

1. Создайте функцию, которая будет корректно выводить сведения об ошибках
2. Найдите в своем приложении места, где могут возникнуть ошибки приложения и сделайте перенаправление на эту функцию, используя оператор *or*.

**Пример выполнения:**

Функция обработки ошибок имеет два аргумента: пользовательское и системное описание ошибки. Перенаправляет приложение на страницу *error.php*, отправляя сведения об ошибке в *GET* запросе:

function handl\_error($error\_message, $system\_error\_message){

header("Location: error.php?" . "error\_message={$error\_message}&" .

"system\_error\_message={$system\_error\_message}");

exit();}

Страница *error.php*

<html lang="ru">

<head>

<meta charset="utf-8" />

<title>Ошибка</title>

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

</head>

<body>

<header>Наша социальная сеть</header>

<h1>Произошла ошибка</h1>

<table><tr>

<td><img src="image/owl.jpg" alt="Умная сова"></td>

<td>Уважаемый, Пользователь!<br>

Наша система не смогла обработать Ваше последнее действие. Мы уже в курсе проблемы и предпримем все возможные действия, чтобы Вас не огорчать.<br>

С уважением, группа поддержки.<br>

Если хотите вернуться назад, то можете <a href="javascript:history.go(-1)">щелкнуть здесь</a><br>

Обратиться лично: forestmarket@yandex.ru

</tr></table><br>

<p>Технические детали:<br></p>

<?php

if (isset($\_GET['error\_message'])){

$error\_message=preg\_replace("/\\\\/", '', $\_GET['error\_message']);}

else {

$error\_message="Вы здесь оказались из-за сбоя в программе.";}

if (isset($\_GET['system\_error\_message'])){

$system\_error\_message=preg\_replace("/\\\\/", '', $\_GET['system\_error\_message']);}

else {$system\_error\_message="Сообщения о системных ошибках отсутствуют";}

echo ("<p>" . $error\_message . "</p>");

echo("<p>Было получено сообщение системного характера:</p>

<b>{$system\_error\_message}</b>");?>

<br><br>

<footer>Ильюшенков Леонид Владимирович</footer>

</body>

</html>

Страница ошибки показана на рис. 20

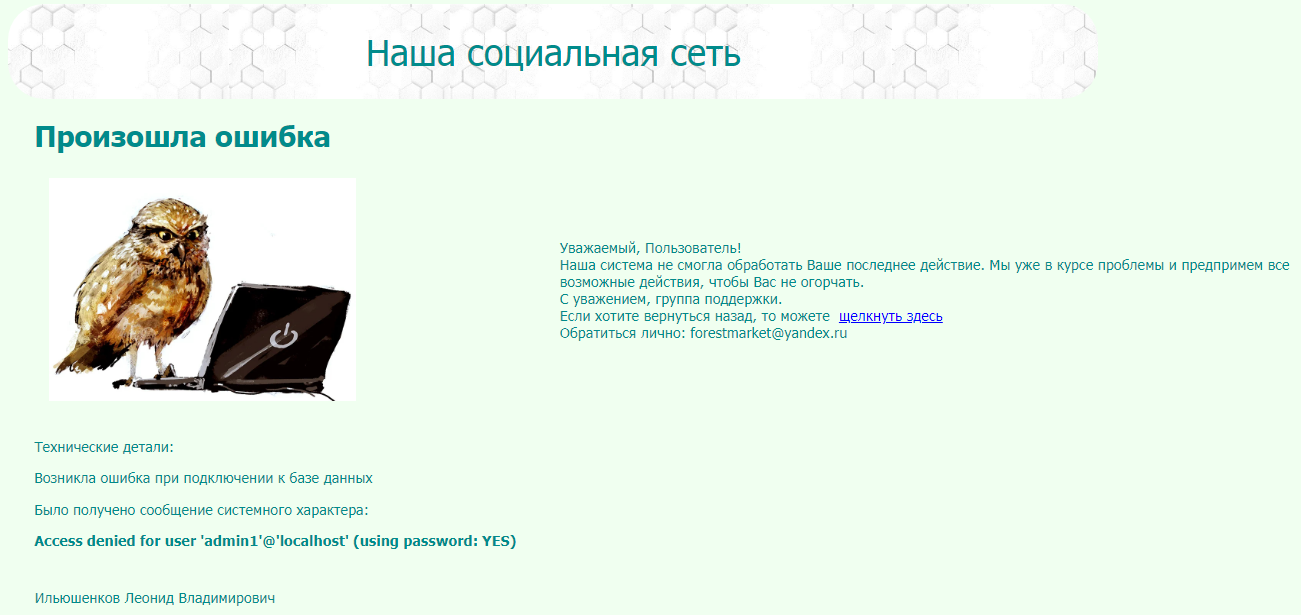


Рис. 20 – Страница отображения ошибок

Самостоятельная работа

1. Изучите функции, содержащие описания ошибок: [www.php.net](http://www.php.net/)

**Практическая работа № 9**

Загрузка пользовательских файлов на сервер

**Теоретическая часть.** Информация о загружаемых через форму файлах содержится в суперглобальной переменной *$\_FILES* — рис. 21

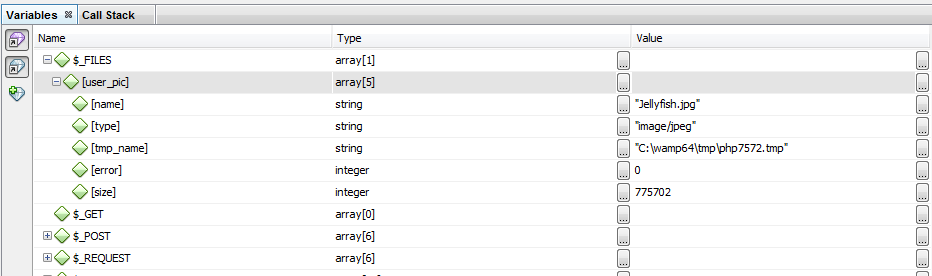


Рис. 21 – Содержание суперглобальной переменной $\_FILES

Содержание этой переменной формируется на стороне пользователя и к ней надо относиться критически. Значение по ключу *error* содержит код ошибки:

Код Описание

1. Нет ошибки
2. Превышен максимальный размер файла, указанный в *php.ini*
3. Превышен максимальный размер файла, указанный форме *HTML*
4. Была отправлена только часть файла
5. Файл для отправки не был выбран

Отправленный файл помещен во временную директорию, указанную в tmp\_name. Файл необходимо переместить на постоянное место хранения, предварительно, убедившись, что мы получили то, что ожидали. Часто задачей злоумышленников является внедрение на сервер файлов с вредоносным кодом и осуществление их запуска. Функция *getimagesize($path)* получает данные об изображении (рис. 22) из файла, загруженного на сервер. Если загруженный файл не является изображением, возникает ошибка. Можно также произвести небольшое сжатие изображения – функция *imagecopyresampled()*. Файлы, не являющиеся изображениями, не сжимаются.

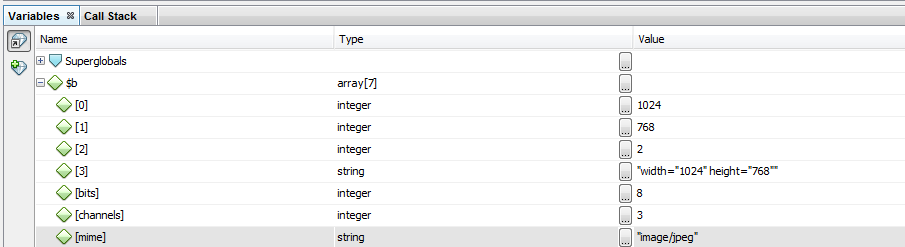


Рис. 22 - Содержание переменной *$b*, полученное при помощи функции *getimagesize($path)*

Для работы с файлами полезны следующие функции:

move\_uploaded\_file ($filename , $destination )

Функция перемещает файл *$filename* в папку назначения *$destination*. Функция возвращает *true* при удачном перемещении.

mkdir($pathname, $mode, $recursive)

Функция создает папку *$pathname*, *$mode* – задает права, по умолчанию – самые широкие права *$mode*=0777; $recursive – разрешает создание вложенных директорий, *$recursive=true/false*.

unlink($filename)

Функция удаляет файл *$filename*.

Файлы каждого пользователя нужно поместить в отдельную папку. Следует продумать системы имен этих папок, таким образом, чтобы по имени можно было понять, к чему относится папка. Тоже самое нужно сделать и с файлами. Чтобы не было ошибок имена файлов и папок не должны повторяться.

Для предотвращения запуска вредоносных скриптов в директории загрузки, необходимо настроить файл *.htaccess*. Для запрета листинга файлов (рис. 23) в папке в файле надо прописать:

Options -Indexes

|  |  |
| --- | --- |
| а) | б) |
|  |  |

Рис. 23 - Обзор содержимого папки в браузере: а – листинг разрешен (по умолчанию); б – листинг запрещен

Для запрета выполнения скриптов в каталоге пропишите:

php\_flag engine 0

AddType "text/html" .php .cgi .pl .fcgi .fpl .phtml .shtml .php2 .php3 .php4 .php5 .asp .jsp

Этим самым мы отключаем PHP в данном каталоге и заставляем все скрипты отображаться как HTML.

**Цель работы.** Освоить безопасную загрузку файлов на сервер

**Задание**

1. Организуйте загрузку файлов пользователя. Для каждого пользователя создайте на сервере свою папку. Продумайте систему имен папок и файлов, так чтобы они не повторялись и было понятно к чему они относятся.
2. Перед отправкой файла на сервер, проверьте наличие ошибок, а также адекватность загружаемых файлов.
3. Внесите путь к файлу в базу данных.
4. Отобразите файл(ы) на странице пользователя.
5. Отключите листинг каталогов в браузере.
6. Отключить выполнение скриптов в каталогах загрузки.

**Пример выполнения.**

<?php

require\_once('app\_config.php');//функция обработки ошибок

$image\_fieldname="user\_pic";//файл загружался с этой меткой в html форме

$php\_errors=array(1 => 'Превышен максимальный размер файла',

2 => 'Превышен максимальный размер файла',

3 => 'Была отправлена только часть файла',

4 => 'Файл для отправки не был выбран');//возможные ошибки

$first\_name= filter\_input(INPUT\_POST, 'first\_name', FILTER\_VALIDATE\_REGEXP,

array('options'=> array('regexp'=>'/(^[A-Za-z]+$)|(^[А-ЯЁа-яё]+$)/u'))) or

$err=true;

$last\_name= filter\_input(INPUT\_POST, 'last\_name', FILTER\_VALIDATE\_REGEXP,

array('options'=> array('regexp'=>'/(^[A-Za-z]+$)|(^[А-ЯЁа-яё]+$)/u'))) or

$err=true;

$email= filter\_input(INPUT\_POST, 'email', FILTER\_VALIDATE\_EMAIL) or $err=true;

$vk= filter\_input(INPUT\_POST, 'vk\_url', FILTER\_VALIDATE\_REGEXP,

array('options'=> array('regexp'=>'/^(vk.com).{3,}$/'))) or $err=true;

$position=strpos($vk, "https://");

if ($position===false){$vk="https://" . $vk;}

//Проверим загрузку файла и является ли он изображением

($\_FILES[$image\_fieldname]['error']==0) or

handl\_error("Сервер не может получить выбранное Вами изображение", $php\_errors[$\_FILES[$image\_fieldname]['error']]);

@is\_uploaded\_file($\_FILES[$image\_fieldname]['tmp\_name']) or

handl\_error("Укажите путь к файлу", "Неопределен путь" . $\_FILES[$image\_fieldname]['tmp\_name']);

$b=@getimagesize($\_FILES[$image\_fieldname]['tmp\_name']) or

handl\_error("Вы выбрали файл, не являющийся изображением", $\_FILES[$image\_fieldname]['tmp\_name'] . "is not image");

$password\_hash= password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

require 'connect.php';

//Проверка существования пользователя с таким же именем

$username= filter\_input(INPUT\_POST, 'username', FILTER\_VALIDATE\_REGEXP,

array('options'=>array('regexp'=>'/^[\w]+$/')));

$stmt= mysqli\_prepare($cn, 'SELECT user\_id FROM student WHERE user\_name=(?)');

mysqli\_stmt\_bind\_param($stmt, 's', $username);

$answ= mysqli\_stmt\_execute($stmt);

mysqli\_stmt\_bind\_result($stmt, $user\_id);

mysqli\_stmt\_fetch($stmt);

if (!$user\_id==null){$err=true;}

if ($err===true){handl\_error('Ошибка загрузки данных', "error=true");}

//есть данные непрошедшие через входной фильтр

$mysqli\_set\_charset = mysqli\_set\_charset($cn, "utf8");

$Date\_reg= time();

//Добавляем пользователя в базу

$stmt= mysqli\_prepare($cn, 'INSERT INTO student(first\_name, last\_name, email, vk\_url, user\_pic\_path, user\_name, password, Date\_reg) VALUES((?), (?), (?), (?), (?), (?), (?), (?))');

mysqli\_stmt\_bind\_param($stmt, 'sssssssi', $first\_name, $last\_name, $email, $vk, $upload\_filename, $username, $password\_hash, $Date\_reg);

$answ= mysqli\_stmt\_execute($stmt) or handl\_error('Ошибка добавления пользователя', mysqli\_stmt\_error($stmt));

$num=strval(mysqli\_stmt\_insert\_id($stmt));

//Создадим имя файла и переместим в папку profile на сервере

$upload\_filename='profile/avatar\_' . $num . '\_' . $\_FILES[$image\_fieldname]['name'];

//имя файла=avatar\_ID\_Имя загруженного файла

@move\_uploaded\_file($\_FILES[$image\_fieldname]['tmp\_name'], $upload\_filename) or

handl\_error("Ошибка перемещения файла", "Ошибка доступа: " . $upload\_filename);

$qery="UPDATE student set user\_pic\_path='{$upload\_filename}' WHERE user\_id={$num}";

$ans= mysqli\_query($cn, $qery);

//Поместим путь к файлу в базу данных

mysqli\_close($cn);

header("Location: show\_user.php?user\_id={$num}"); ?>//перенаправление на показ пользователя

**Вопросы для самостоятельной работы**

1. Изучите другие настройки .htaccess

<https://ru.hostings.info/schools/htaccess.html>

1. Изучите организацию хранения файлов в базе данных *Mysql* в виде двоичных объектов *blob*.

**Практическая работа № 10**

Динамическое обращение к серверу, технология *ajax*

**Теоретическая часть.** Для сетевых запросов из *JavaScript* есть широко известный термин «*AJAX*» (аббревиатура от *Asynchronous JavaScript And XML*). Наиболее мощным и современным является метод *fetch*, с базовым синтаксисом:

let promise = fetch(url, [options])

где *url* – адрес для отправки запроса; *options* – дополнительные параметры: метод, заголовки и так далее. *Promise* выполняется с объектом встроенного класса *Response* в качестве результата, как только сервер пришлёт заголовки ответа. Промис завершается с ошибкой, если *fetch* не смог выполнить *HTTP*-запрос, например, при ошибке сети или если нет такого сайта. *HTTP*-статусы 404 и 500 не являются ошибкой. Мы можем увидеть *HTTP*-статус в свойствах ответа:

* *status* – код статуса *HTTP*-запроса, например 200;
* *ok* – логическое значение: будет *true*, если код HTTP-статуса в диапазоне 200-299.

*Response* предоставляет несколько методов, основанных на промисах, для доступа к телу ответа в различных форматах:

* *response.text()* – читает ответ и возвращает как обычный текст;
* *response.json()* – декодирует ответ в формате *JSON*;
* *response.formData()* – возвращает ответ как объект *FormData*.

Пример *POST* запроса отправки изображений:

async function ajx(){

a=document.getElementById('file');//input c файлом

b=a.files;

formData = new FormData(); // новая форма для передачи на сервер

formData.append('file\_name', a[0]['name']);//Добавить данные в форму

formData.append('mime\_type', a[0]['type']);

formData.append('file\_size', a[0]['size']);

id\_u=document.getElementById('id\_u');

id\_u=id\_u.innerText; //id пользователя (хозяина изображения)

formData.append('profile\_pic\_id', id\_u);

formData.append("image\_data", a.files[0]);

parag=document.getElementById('itog'); //окошко для вывода результата запроса

try {

const response = await fetch('another\_pic.php', {// попытаться выполнить запрос

method: 'POST',

body: formData

});

const result = await response.json(); //удачно

console.log('Успех:', JSON.stringify(result));

parag.innerHTML='Изображения загружены';

}

catch (error) { //неудано

console.log('Ошибка:', error);

parag.innerHTML='Ошибка загрузки';}

}

}

Для тестирования и составления запросов удобно использовать программу *Postman* (<https://www.postman.com/downloads/>). Составление запроса к сайту для добавления блюда:

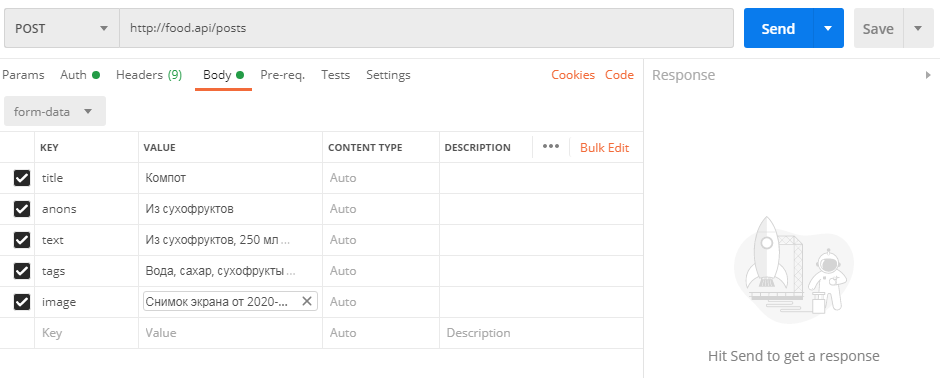


Рис. 24 – Запрос в программе *Postman*

Программный код отправки запроса – рис. 25

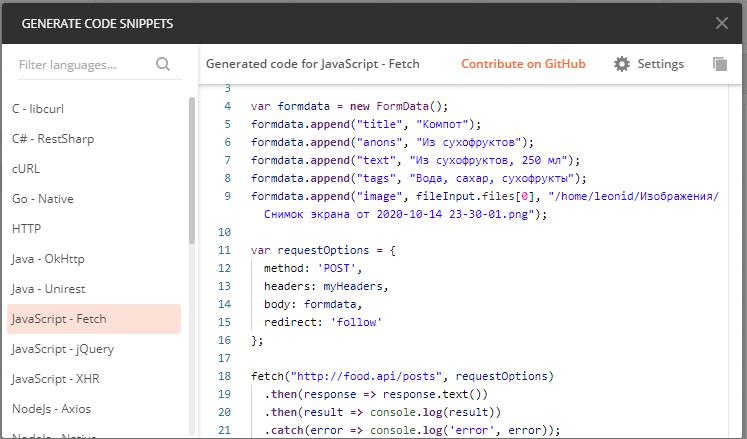


Рис. 25 – Программный код запроса

Отправка ответа в формате *json* на сервере производиться:

echo json\_encode($answer);

Отправка статуса запроса:

http\_response\_code(200);

Список кодов – статусов запросов:

<https://developer.mozilla.org/ru/docs/Web/HTTP/Status>

**Цель работы:** Создание динамических запросов к серверу

**Задание.**

1. При регистрации произведите проверку наличия пользователя и(или) электронной почты с таким именем;
2. Составьте еще 1..3 запроса исходя из функционала проектируемого сайта.

**Пример выполнения.**

Проверка наличия пользователя с таким же логином в базе (клиентская часть):

async function uniq\_username(username, valid) {

form=new FormData();

form.append('username', username);

try {

const response = await fetch('testusername.php', {

method: 'POST',

body: form

});

let result = await response.text();

console.log('Успех:', result);

a=document.getElementById('form');

if (valid){

a.rows[8].cells[2].innerHTML='<span class="fa fa-check"></span>';

}

else {

a.rows[8].cells[2].innerHTML='<span class="fa fa-times"></span>';

}

controll();

if (result=='false') {

a.rows[8].cells[2].innerHTML="<span class='fa fa-times'>\n\

Пользователь с таким именем уже существует</span>";

}

} catch (error) {

console.log('Ошибка:', error);

a.rows[8].cells[2].innerHTML='<span class="fa fa-times"></span>';

}

}

Серверная часть (файл *testusername.php*)

<?php

$username= filter\_var(trim($\_POST['username']), FILTER\_VALIDATE\_REGEXP,

array('options'=>array('regexp'=>'/^[\w]+$/')));

require 'connect.php';

$stmt= mysqli\_prepare($cn, 'SELECT user\_id FROM student WHERE user\_name=(?)');

mysqli\_stmt\_bind\_param($stmt, 's', $username);

$answ= mysqli\_stmt\_execute($stmt);

mysqli\_stmt\_bind\_result($stmt, $user\_id);

mysqli\_stmt\_fetch($stmt);

if ($user\_id) {

$result='false';

}

else {

$result='true';

}

echo $result;?>

Результат работы программы представлен на рис. 26

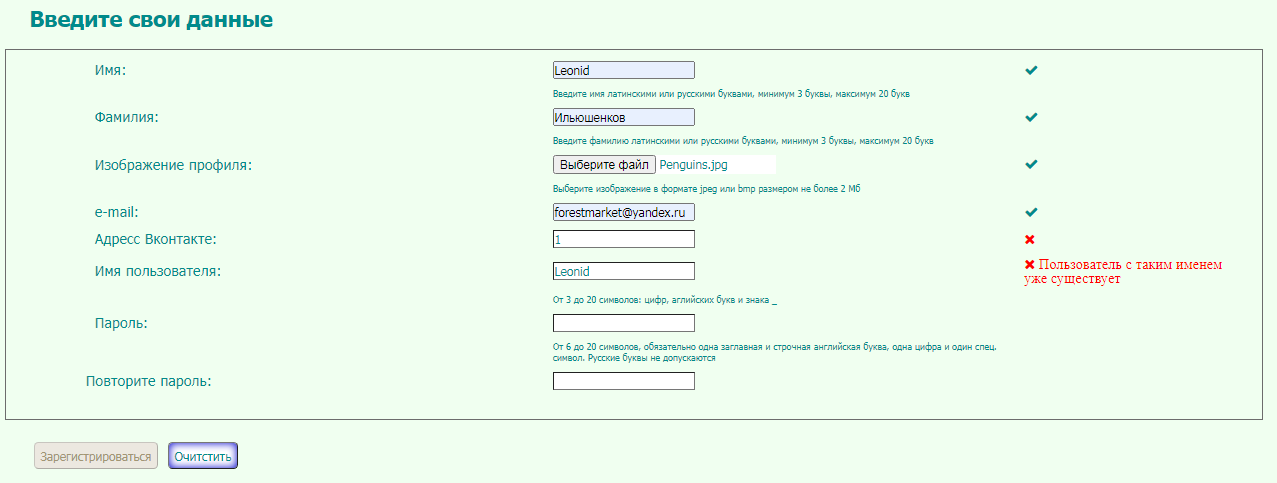


Рис. 26 – Проверка уникальности имени пользователя

**Практическая работа № 11**

Работа с электронной почтой

**Теоретическая часть.** Для отправки почты используется приложение *sendmail*, входящее в пакет *OpenServer*. Это приложение необходимо настроить – рис. 27. Данные *STMP* и *POP3* сервера предоставляет провайдер хостинга, или можно взять со стороннего сервиса отправки сообщений.

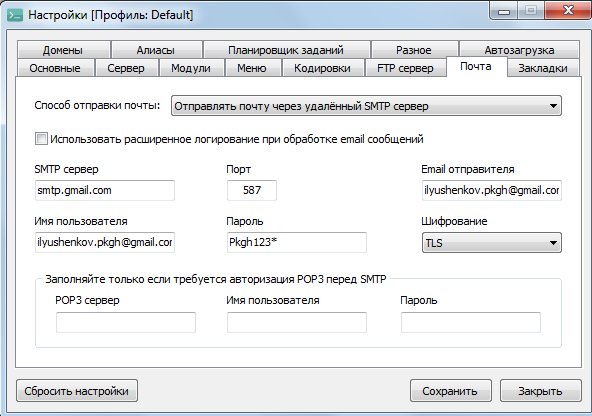


Рис. 27 – Пример настройки почтовых программ

Компания *Google* считает *sendmail* ненадежной программой, для этого в настройках ей надо дать разрешение (Акаунт – Настройки – Безопасность – Ненадежные приложения, у которых есть доступ к аккаунту, рис. 28).

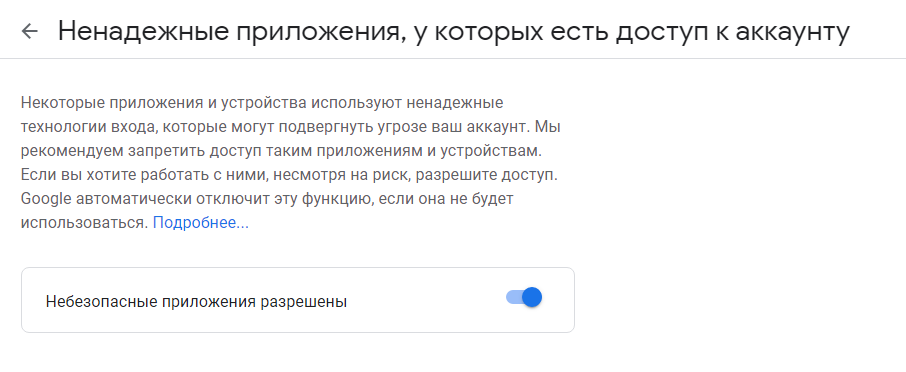


Рис. 28 – Настройка доступа к аккаунту *sendmail*

Для отправки почты используют функцию *mail*

mail($to, $subject, $message, $additional\_headers)

где *$to* – получатель(ли) письма; *$subject* – тема письма; *$message* – текст письма; *$additional\_headers* – дополнительные заголовки письма (необязательно). Сведения об ошибках отправки письма: *~\OpenServer\userdata\logs*.

**Цель работы.** Освоить навыки отправки писем, подтверждения адреса электронной почты при регистрации.

**Задание.**

1. После внесения нового пользователя в базу данных, следует отправить на адрес электронной почты письмо с подтверждением правильности адреса
2. При клике по ссылке в письме необходимо в базе данных отметить адрес, как подтвержденный
3. Необходимо убрать со страницы пользователя предупреждение о необходимости подтвердить адрес электронной почты
4. Отправить и обработать письмо на изменение пароля, в случае если пользователь его забыл

**Пример выполнения:**

$subject='Регистрация на сайте «Наша социальная сеть»';

$massege="<html>

<head>

<title>Регистрация на сайте «Наша социальная сеть»</title>

<style type='text/css'>

\*{

background-color: #F0FFF0;

font-family: Tahoma;

color: #008B8B;

padding-left: 2%;

}

</style>

</head>

<body>

<h1>Добро пожаловать на сайт «Наша социальная сеть»!</h1>

<p>Для того чтобы завершить процесс регистрации перейдите по ссылке:

<a href='http://student.rus/check\_email.php?email={$email}&massage={$Date\_reg}'>

Подтвердить почту</a></p>

<footer>С уважением, администрация, student.rus</footer>

</body>

</html>";

$headers = "Content-type: text/html; charset=iso-8859-1\r\n";

$headers .= "From: Administrator <ilyushenkov.pkgh@gmail.com>\r\n";

$mail=mail($email, $subject, $massege, $headers);

header("Location: show\_user.php"); ?>

Пример письма см. рис. 29

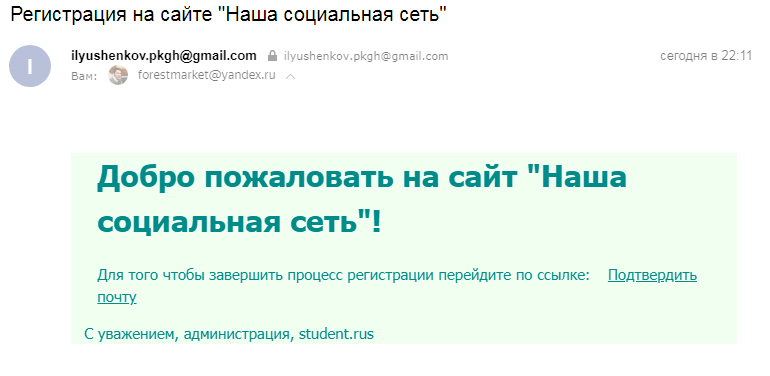


Рис. 29 – Письмо подтверждение электронной почты пользователя

Письмо содержит *GET* запрос – http://student.rus/check\_email.php?email [=forestmarket@yandex.ru&message=1603566708](mailto:=forestmarket@yandex.ru&message=1603566708). Обработка запроса:

<html lang="ru">

<head>

<meta charset="utf-8" />

<title>Регистрация студентов</title>

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

<script type="text/javascript"src="main1.js"></script>

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

</head>

<body>

<header>Наша социальная сеть</header>

<?php

$mail= filter\_input(INPUT\_GET, 'email', FILTER\_VALIDATE\_EMAIL);

$message= filter\_input(INPUT\_GET, 'massage', FILTER\_VALIDATE\_INT);

require 'connect.php';

$stmt= mysqli\_prepare($cn, 'SELECT user\_id FROM student WHERE email=(?) AND Date\_reg=(?)');

mysqli\_stmt\_bind\_param($stmt, 'sd', $mail, $message);

mysqli\_stmt\_execute($stmt);

mysqli\_stmt\_bind\_result($stmt, $id);

mysqli\_stmt\_fetch($stmt);

if($id){

$stmt='';

$stmt= mysqli\_prepare($cn, 'UPDATE student SET check\_emial=1 WHERE user\_id=(?)');

mysqli\_stmt\_bind\_param($stmt, 'd', $id);

$var=mysqli\_stmt\_execute($stmt);

exit("<h3>Поздравляем!!!</h3>

<p>Электронный адрес успешно подтвержден!</p>

<a href='show\_user.php'>Перейти на свою страницу</a>

<footer>Ильюшенков Леонид Владимирович</footer>");

}

else {

exit("<p>Нам очень жаль! Ошибка подтверждения адреса электронной почты</p>

<a href='show\_user.php'>Перейти на свою страницу</a>

<footer>Ильюшенков Леонид Владимирович</footer>");

}

mysqli\_close($cn);

?>

</body>

</html>

Результат работы скрипта представлен на рис. 30.

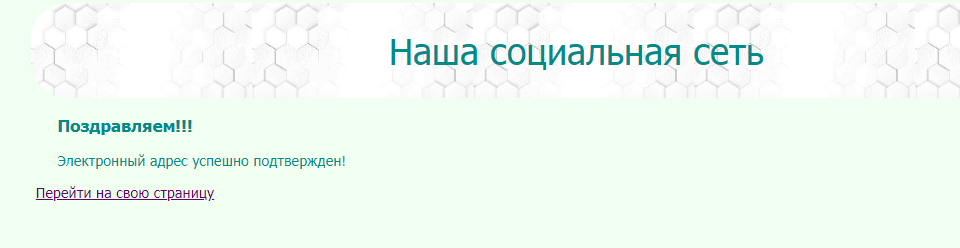
****

Рис. 30 – Подтверждение регистрации

На личной странице пользователя следует убрать сообщение о необходимости подтверждения электронной почты.

**Вопросы для самостоятельной работы**

1. Освойте отправку электронных писем с вложениями:

<https://habr.com/ru/post/444744/>

1. Освойте другие способы отправки писем: *mb\_send\_mail(); imap\_mail(); PEAR::Mail*.

**Практическая работа № 12**

Аутентификация с использованием cookie-файлов

**Теоретическая часть.** Технология cookie – это средство позволяющее хранить отдельные фрагменты информации на стороне клиента. У cookie-файла есть имя, значение, являющиеся единым информационным фрагментом и дата истечения годности. Значения cookie можно извлечь из суперглобальной переменной $\_COOKIE, представляющий собой ассоциативный массив. Для установки cookie используют функцию:

setcookie('key', 'value', 'time');

где *key* – ключ к элементу ассоциативного массива, *value* – значение, *time* – время действия cookie с момента начала эры в системах UNIX и Linux (с 00.00 01.01.1970 по Гринвичу). Сколько прошло с этого момента определяет функция time(), к этому времени следует прибавить продолжительность жизни cookie в секундах. Например:

setcookie('id', 26, time()+3600);

Установить в файле cookie по ключу *id* значение 26 в течение 1 часа (3600 с). Параметр *time* может быть не установлен. В этом случае coockie живет до закрытия браузера.

Для отмены действия cookie параметр *time* задают как прошедший, например:

setcookie('id', 26, time()-3600);

**Цель работы**: Освоить аутентификацию при помощи технологии cookie.

**Задание.**

1. При авторизации и регистрации создайте файл cookie, содержащий

По ключу id – номер id;

По ключу token – сгенерированный произвольный токен.

1. Если при авторизации установлена галочка «Оставаться на сайте», следует обеспечить работоспособность cookie в течение 2 месяцев;
2. Cookie должны отменяться при нажатии на «Выйти из приложения».
3. Создайте на заглавной странице сайта предупреждения о использовании технологии cookie.
4. При нажатии «Выйти из приложения» отмените cookie.

**Пример выполнения**

//установка cookie при создании пользователя

//………

// запись в базу

require 'connect.php';//подкдючение к базе данных

$Date\_reg=time();

$token=bin2hex(random\_bytes(15));//сгенерируем токен

$stmt= mysqli\_prepare($cn, 'INSERT INTO student(first\_name, last\_name, email, vk\_url,

user\_name, password, Date\_reg, token) VALUES((?), (?), (?), (?), (?), (?), (?), (?))');

mysqli\_stmt\_bind\_param($stmt, 'ssssssis', $first\_name, $last\_name, $email, $vk, $username, $password\_hash, $Date\_reg, $token);

$answ= mysqli\_stmt\_execute($stmt) or handl\_error('Ошибка добавления поль-зователя', mysqli\_error($cn));//поместим токен с остальными данными в базу дан-ных

$num=strval(mysqli\_stmt\_insert\_id($stmt));//узнаем Id пользователя

setcookie('user\_id', $num);//установим coockie

setcookie('token', $token);

mysqli\_close($cn);

Функция проверки cookie

//функция проверяет данные cookie.

//В случае успешной проверки возвращает значение id пользователя

//В случае неуспешной проверки возвращает false

function check\_cookie(){

require 'connect.php';

$user\_id= filter\_input(INPUT\_COOKIE, 'user\_id', FILTER\_VALIDATE\_INT);

$token= filter\_input(INPUT\_COOKIE, 'token', FILTER\_VALIDATE\_REGEXP,

array('options'=>array('regexp'=>'/^[0-9a-f]+$/')));

$query="SELECT token FROM student WHERE user\_id='{$user\_id}'";

$result= mysqli\_query($cn, $query);

$row= mysqli\_fetch\_array($result);

if($token==$row['token']){

return $user\_id;

}

else {

return false;

}

}

Работа с cookie при авторизации

<?php

$user\_name= filter\_input(INPUT\_POST, 'username', FILTER\_VALIDATE\_REGEXP,

array('options'=>array('regexp'=>'/^[\w]+$/')));

$password=$\_POST['password'];

require 'connect.php';

$stmt= mysqli\_prepare($cn, "SELECT user\_id, password FROM student WHERE user\_name=(?);");

mysqli\_stmt\_bind\_param($stmt, 's', $user\_name);

$answ= mysqli\_stmt\_execute($stmt);

mysqli\_stmt\_bind\_result($stmt, $user\_id, $password\_hash);

$result=mysqli\_stmt\_fetch($stmt);

mysqli\_stmt\_close($stmt);

if (isset($user\_id) && password\_verify($\_POST['password'], $password\_hash)){

//При успешной авторизации в базу данных помещаем token и устанавливаем cookie

$token= bin2hex(random\_bytes(15));

$query="UPDATE student SET token='{$token}' WHERE user\_id='{$user\_id}';";

$row=mysqli\_query($cn, $query) or handl\_error('Ошибка авторизации', mysqli\_error($cn));

setcookie('user\_id', $user\_id);

setcookie('token', $token);

if ($\_POST['stay']){

//Если установлена галочка «Оставаться на сайте», действие cookie продлеваем на 60 дней

setcookie('user\_id', $user\_id, time()+3600\*24\*60);

setcookie('token', $token, time()+3600\*24\*60);

}

header("Location: show\_user.php");

}

else {

//Отобразим ошибку на странице в случае ошибки авторизации

header("Location: index.php?error=1");

}

Проверка cookie на всех страницах, где это необходимо

require 'connect.php';

require\_once 'check\_cookie.php';

$user\_id= check\_cookie();

if (!$user\_id){

header("Location: index.php");}

Отмена cookie, нажатие «Выйти из приложения»

setcookie('user\_id', '', time()-2\*3600\*24\*182);

setcookie('token', '', time()-2\*3600\*24\*182);

header("Location: index.php");