Санкт-Петербургский губернаторский физико-математический лицей №30

Летняя учебно-исследовательская практика 2016 года

Группа 16. "Web программирование".

Мартьянов Сергей
10-5 класс ФМЛ №30

Санкт-Петербург
1-17 июня **2016** года

Содержание

1 Введение

Летняя учебно-исследовательская практика была посвещена изучению Web-программирования и проходила в ФМЛ №30 в период с 01 июня по 17 июня.

Изучены и освоены темы:

- Шаблоны на Web-сайтах
- Принцип работы сессий на Web-сайтах
- Особенности использования PostgreSQL
- Использование CSS на Web-сайтах
- Особенности издательской системы LaTeX

За время проходения практики было создано web-приложение - таблица умножения, требующее авторизации и имеющее администраторскую панель. Скриншоты можно посмотреть в приложении 8.10.

Для решение этой задачи, есть несколько вариантов:

- **Использование статических страниц**

  Единственным достоинством данного способа является простота реализации. Но против этого сомнительного достоинства есть уйма недостатков: невозможность отделение данных от представления, сложность обновления содержимого сайта и взаимодействия с пользователем и т.д.

- **Использование CGI скриптов**

  Этот способ позволяет обрабатывать данные на стороне сервера и отправлять их клиенту, при этом за счет использования низкоуровневых языков программирования (например Си) скорость этой обработки выше, чем скорость обработки данных средствами интерпритируемых языков программирования.

  Но использование этого способа требует много ресурсов, так что при большом количестве подключений к серверу, его работоспособность может нарушиться.

  Пример CGI-скрипта на языке C можно увидеть в приложении 8.8.

- **Использование интерпритируемых языков программирования**

  При использовании этого способа, нагрузка на сервер будет меньше, чем при использовании CGI-скриптов, но и обработка донных будет проходить медленее. На мой взгляд, основным преймуществом этого способа является то, что интерпритируемые языки программирования предоставляют много встроенных функций для взаимодействия сервера и клиянта.

Был выбран последний способ, т.к. он показался самым оптимальным для данной задачи. На сервере использовался язык PHP, хранения данных была выбрана СУБД PostgreSQL.

2 Создание HTML-шаблона

Первым шагом в реализации проекта было создание шаблонов на языке *HTML* и программы на языке *PHP*, которая в зависимости от ситуации выводит на экран пользователя тот или иной HTML-шаблон, подставляя туда необходимые данные. Таким образом было реализовано разделение данных от их представления, что существенно упрощает поддержку сайт в дальнейшем, так как не придется менять представление при добавлении новых данных, а при изменении представления, данные не будут потеряны или повреждены.

В приложении 8.1 представлен класс HTML, который представляет из себя набор переменных, описывающих некоторые характеристики сайта, например, название и набор методов, которые генерируют HTML код. А в приложении 8.2 приведен пример использования этого класса на странице index.php.

3 Реализация сессий

Сессии необходимы на сайте, чтобы хранить информацию о пользователе за текущий сеанс. В языке PHP есть стандартное средство работы с сессиями, но на у нас практике было реализован свой механизм, с использованием технологии *Cookies*. Такой подход был выбран, чтобы понять принцип работы сессий. В нашем проекте в сессии хранились данные о текущем пользователе.

Для этого был создан класс Session, который можно увидеть в приложении 8.3.

4 Подключение к СУБД

После сохранения данных о текущем пользователе был вопрос, где хранить данные о всех пользователей и как и в какой момент их записывать в сессию, то есть необходимо было сделать механизм авторизации. Изначатьно данные о пользователях хранились в одном из файлов в корне нашего проекта, но хранились там только логин и пароль, если хранить там еще другие данные, текст в файле будет иметь сложную структуру и для выборки определенных данных из этого файла пришлось бы писать сложные функции на PHP.

Для решения этой проблемы использовалась *система управления базами данных (СУБД)* **PostgreSQL**. Она была выбрана из-за своих особенностей работы в процедурами, поддержки помима языка SQL своего языка программирования plpgsql и удобной интеграции с языком PHP. В базе данных кроме логина и пароля пользователей могли храниться и другие данные, такие как его email или, когда он сделал последний вход.

Для соединения с БД создан отдельный класс DB (см. приложение 8.4). При работе с БД кроме функций на PHP использовались функции, непосредственно хранящиеся в СУБД, написанные на SQL и plpgsql, их можно увидеть в приложении 8.5. Эти функции нужны для работы с пользователями.

Вимение! Данные функции работают только в PostgreSQL!!!

5 Разграничение доступа

Когда на сайте появилось много пользователей, необходима была администраторская панель, чтобы можно было видеть список этих пользователей и при необходимости заблокировать кого-либо или же поправить его профиль. Но к этой панели доступ должен был быть открыт только определенным пользователем. Чтобы это реализовать, была создана система ролей — каждый пользователь имеет свою роль, а каждая роль свои привелегии. Таким образом администратор может редактировать свой и чужие профили, пользователь — только свой, а гость, соответсвенно, никакой профиль редактировать не может. Эта система удобна тем, что если будет потребность создать новую группу пользователей с новыми правами, не нужно будет координально менять таблицу пользователей.

Для текущего пользователя был создан класс User, который является "посредником"между классами DB и Session. Его код в приложении 8.6, а код схемы базы данных в приложении 8.7.

6 Создание таблицы стилей

Последним этапом в разработке проекта было его оформление. Для этого достаточно было просто создать файл *CSS* и подключить его в заголовке. На этом этапе очень ощутимо было использование шаблонов, т.к. фактически было изменение представления данных, и если бы оно было тесно связано с данными и "логикой"сайта, пришлось бы менять каждую страницу, а в нашем случае были изменены лишь несколько шаблонов.

Таблицы стилей подгружаются автоматически (см. приложение 8.1).

Самое интересное, что было реализовано — небольшая анимация на языке CSS. Ее код можно найти в приложении 8.9.

7 Заключение

В течении практики было создано простое Web-приложение, но оно использовало довольно много возможностей языка PHP и СУБД PostgreSQL. Я извлек для себя немало полезного: получил опыт работы в PostgreSQL, изменил представление о структуре Web-проекта, о разделении данных от шаблонов, написанные шаблоны смогут пригодиться для дальнейших проектов.

Еще до прохождения практики я использовал сессию, но не задумывался, как она работает, а сейчас я могу сам реализовать такой механизм. Также я узнал полезные вещи о работе с Linux, о CGI и про CSS конвертер Sass. Считаю, что практика принесла мне большую пользу.

8 Приложения

8.1 html.php

```
<?php
```

```
define('TYPE_TEMPLATE', 0);
define('TYPE_META', 1);

class HTML
{
    static public $SiteName = "MySite";
    static $templates = [];
    static $js = [];
    static $css = [];
    static public $templates_dir = "templates/";

    static public function header( $title = "" ) //Генерация заголовка
    {
        if ($title == "")
            $title = HTML::$SiteName;
        else
            $title = HTML::$SiteName . " - " . $title;

        HTML::template("header", array($title));
    }

    static public function footer() //Генерация "подвала"
    {
        HTML::template("footer");
    }

    static public function template($name, $args = array())
    //Добавление в массив шаблонов необходимый html файл
    {
        if (file_exists(HTML::$templates_dir . $name . ".meta.php"))
            HTML::$templates[] = array($name . ".meta", array(), TYPE_META);
        HTML::$templates[] = array($name, $args, TYPE_TEMPLATE);
    }

    static function flush() //Печать шаблонов на монитор
    {
        foreach (HTML::$templates as $t) //Шаблоны с мета-данными
        {
            list($name, $args, $type) = $t;
            if ($type != TYPE_META)
                continue;
            include(HTML::$templates_dir . $name . ".php");
        }
        foreach (HTML::$templates as $t)
        {
            list($name, $args, $type) = $t;
            if ($type == TYPE_META)
                continue;
            include(HTML::$templates_dir . $name . ".php");
        }
```

```
    }

    static function include_js($name) //Сбор необходимы JS файлов
    {
        HTML::$js[] = $name;
    }

    static function put_js()
    {
        HTML::$js = array_unique(HTML::$js); //Печать JS файлов
        foreach (HTML::$js as $name)
            echo "<script type='text/javascript' src='js/" . $name . "'></script>";
    }

    static function include_css($name) //Сбор необходимы CSS файлов
    {
        HTML::$css[] = $name;
    }

    static function put_css() //Печать CSS файлов
    {
        HTML::$css = array_unique(HTML::$css);
        foreach (HTML::$css as $name)
            echo "<link rel='stylesheet' href='css/" . $name . "' />";
    }
}

?>
```

8.2 index.php

```
<?php

require_once("html.php");

HTML::header("Главная"); //Подключение заголовка
HTML::template("index"); //Подключение нужной страницы
HTML::footer();          //Подключение "подвала"

HTML::flush();           //Вывод всего подключенного в буфер

/*

Таким образом страница index.php "достает" код из файлов header.php,
index.php, footer.php из дирекотрии temlates, объединяет его и выводит
на экран. Если нужно сделать какие-то операции на языке PHP, они делаются
перед поключением этих шаблонов, а потом просто передаются их результаты
в качестве параметров, и чтобы изменить представление этих параметров,
достаточно изменить файл index.php в директории templates, при этом не
затрагивая сами операции на PHP.
```

```
*/

?>
```

8.3 session.php

```
<?php

class Session
{
    static public $cookie_name = "SPR2016";
    static public $session_dir = "sessions/";
    static public $data = array();
    static public $started = false;
    static public $uid = "";

    static public function start_session() //Запуск сессию
    {
        if (isset($_COOKIE[Session::$cookie_name]))
            Session::restore_session();
        else
        {
            Session::$uid = md5(microtime(true));
            setcookie(Session::$cookie_name, Session::$uid);
        }
        Session::$started = true;
    }

    static public function store_session() //Сохранение сессии
    {
        if (!Session::$started)
            return;
        if (file_put_contents(Session::$session_dir . Session::$uid,
        serialize(Session::$data)) === false)
            die("Couldn't save session data.");
    }

    static public function restore_session() //Загрузка сессии
    {
        Session::$uid = $_COOKIE[Session::$cookie_name];
        Session::$data = unserialize(file_get_contents(Session::$session_dir .
        Session::$uid));
    }

    static public function set($name, $value) //Добавление парамерта в сессию
    {
        Session::$data[$name] = $value;
    }
```

```
    static public function get($name) //Получение параметра из сессии
    {
        if (!isset(Session::$data[$name]))
            return null;
        return Session::$data[$name];
    }
}

Session::start_session();

/*
На каждой странице в начале должна запускаться сессия. Если пользователь зашел
первый раз за сеанс, ему дается кука, иначе по этой куке с файла сервера
загружается информация об этой сессии. Сохранение сессии в файл на сервере
следует делать после какого-либо изменения каких либо-данных сессии.
*/

?>
```

8.4 db.php

```
<?php

class DB
{
    private $conn = null;

    function __construct($conn_string = "") //Подключение к СУБД PostgreSQL
    {
        if ($conn_string == "")
            return;

        if (($this->conn = pg_connect($conn_string)) === false)
            die("Не удалось подключиться к СУБД.");

    }

    function __destruct() //Отключение от СУБД PostgreSQL
    {
        if ($this->conn !== null && $this->conn !== false)
            pg_close($this->conn);
    }

    function one_row($resource) //Получение одной строки по ресурсу
    {
        if ($resource === false)
            die(pg_last_error($this->conn));
        $res = pg_fetch_assoc($resource);
        return ($res === false) ? null : $res;
    }
```

```
function all_rows($resource) //Получение всех строк из ресурса
{
    if ($resource === false)
        die(pg_last_error($resource));
    $res = [];
    while (($row = pg_fetch_assoc($resource)) !== false)
        $res[] = $row;
    return $res;
}

//Следующие функции получают из БД какую-любо информацию о пользователе
//Они используют SQL-функции на стороне СУБД, о которых есть в других
//приложениях

function check_auth($login, $passwd)
{
    $resource = pg_query_params($this->conn, "SELECT * FROM auth($1, $2)",
        array($login, md5($passwd))) or die(pg_last_error());
    return $this->one_row($resource);
}

function update_profile($id, $login, $passwd, $name, $email, $disabled)
{
    $resource = pg_query_params($this->conn,
        "SELECT user_update($1, $2, $3, $4, $5, $6)",
        array($id, $login, ($passwd) ? md5($passwd) : null,
        $name, $email, $disabled)) or die(pg_last_error());
    return pg_fetch_result($resource, 0, 0);
}

function create_profile($login, $passwd, $name, $email, $reg_date)
{
    $resource = pg_query_params($this->conn, "SELECT user_create($1, $2, $3, $4,
    $5)", array($login, ($passwd) ? md5($passwd) : null, $name, $email, $reg_date))
    or die(pg_last_error());
    return pg_fetch_result($resource, 0, 0);
}

function get_users($id = null)
{
    if ($id) {
        $resource = pg_query_params($this->conn, "SELECT * FROM get_users($1)",
            array($id)) or die(pg_last_error());
        return $this->one_row($resource);
    }
    else
        $resource = pg_query($this->conn, "SELECT * FROM get_users(null)")
        or die(pg_last_error());
    return $this->all_rows($resource);
```

```
    }

    function get_rights($id)
    {
        $resource = pg_query_params($this->conn, "SELECT * FROM get_rights($1)",
            array($id)) or die(pg_last_error());
        return $this->one_row($resource);
    }
}

$db = new DB("dbname=mult user=mult");

?>
```

8.5 users.sql

```
--Вывод пользователя по id, если id не передан, то выводит все пользователей
CREATE OR REPLACE FUNCTION get_users( id int )
  RETURNS TABLE (
    id int,
    login text,
    name text,
    email text,
    reg_date date,
    last_login timestamp,
    disabled bool
  ) AS $$
    SELECT id, login, name, email, reg_date, last_login, disabled FROM users
    WHERE CASE $1 IS NULL
      WHEN TRUE THEN TRUE
      ELSE id=$1
    END
    ORDER BY reg_date, id;
  $$ LANGUAGE SQL STABLE;

--Создание нового пользователя
CREATE OR REPLACE FUNCTION user_create(
login text, passwd text, name text, email text, reg_date date
)
  RETURNS int AS $$
    DECLARE
      i int;
    BEGIN
      INSERT INTO users (login, passwd, name, email, reg_date, last_login)
      VALUES ($1, $2, $3, $4, $5, NOW())
      RETURNING id INTO i;
      RETURN i;
    EXCEPTION
      WHEN UNIQUE_VIOLATION THEN
        RETURN -1;
```

```
      WHEN NOT_NULL_VIOLATION THEN
        RETURN -2;
    END;
  $$ LANGUAGE plpgsql VOLATILE;

--Редактирование пользователя
CREATE OR REPLACE FUNCTION user_update(
id int, login text, passwd text, name text, email text, disabled bool
)
  RETURNS int AS $$
    DECLARE
      i int;
    BEGIN
      UPDATE users SET (login, passwd, name, email, disabled) = ($2,
        CASE $3 IS NULL
          WHEN TRUE THEN (SELECT users.passwd FROM users WHERE users.id=$1)
        ELSE $3
        END, $4, $5,
        CASE $6 IS NULL
          WHEN TRUE THEN FALSE
          ELSE $6
        END
      )
      WHERE users.id=$1
      RETURNING $1 INTO i;
      RETURN i;
    EXCEPTION
      WHEN UNIQUE_VIOLATION THEN
        RETURN -1;
      WHEN NOT_NULL_VIOLATION THEN
        RETURN -2;
    END;
$$ LANGUAGE plpgsql VOLATILE;

--Авторизация пользователя
CREATE OR REPLACE FUNCTION auth( l_login text, l_passwd text )
  RETURNS TABLE (
    id int,
    login text,
    name text,
    email text,
    reg_date date,
    last_login timestamp
  ) AS $$
    DECLARE
      l_last_login TIMESTAMP;
    BEGIN
      SELECT users.last_login INTO l_last_login FROM users
      WHERE users.login=$1 AND users.passwd=$2 AND NOT users.disabled;
      UPDATE users SET last_login=NOW() WHERE users.login=$1 AND users.passwd=$2
```

```
      AND NOT users.disabled;
      RETURN QUERY
        SELECT users.id, users.login, users.name,
        users.email, users.reg_date, l_last_login
        FROM users WHERE users.login=$1 AND users.passwd=$2 AND NOT users.disabled;
    END;
  $$ LANGUAGE plpgsql VOLATILE;

--Получение прав пользователя
CREATE OR REPLACE FUNCTION get_rights( id int )
  RETURNS TABLE (
    profile_upd bool,
    users_upd bool
  ) AS $$
    SELECT profile_upd, users_upd FROM roles
    WHERE id=(SELECT role_id FROM users WHERE id=$1);
$$ LANGUAGE SQL STABLE;
```

8.6 user.php

```
<?php

require_once("session.php");
require_once("db.php");

class User
{
private $authenticated;
    private $profile;
    static $session_key = "user_info";
    //Параметр сессии, который хранит информацию о пользователе

    function __construct()
    {
        $this->get_from_session(); //Загрузка из сессии
    }

    function __destruct()
    {
        $this->set_session(); //Передача в сессию
    }

function get_from_session()
{
if (!Session::$started)
        {
            $this->fill_unauth();
            return false;
        }
```

```
        if (($sess_data = Session::get(User::$session_key)) === null)
        {
            $this->fill_unauth();
            return false;
        }

        $this->authenticated = $sess_data['authenticated'];
        $this->profile = $sess_data['profile'];

        return $this->authenticated;
}

function set_session()
{
Session::set(User::$session_key,
            array('authenticated' => $this->authenticated, 'profile' => $this->profile));
}

    //Заполнить данные о пользователе, как о неавторизованным
    function fill_unauth()
    {
        $this->authenticated = false;
        $this->profile = array(
            "rights" => array("profile_upd" => false, "users_upd" => false)
        );
    }

    //Авторизация пользователя
    function authorize($login, $password)
    {
        global $db;

        if (($this->profile = $db->check_auth($login, $password)) !== null)
        {
            $this->authenticated = true;
            $this->profile['rights'] = $db->get_rights($this->profile['id']);
            $this->set_session();
            return true;
        }

        $this->fill_unauth();
        return false;
    }

    //Обновление профиля
    function update_profile($id, $login, $passwd, $name, $email, $disabled)
    {
        global $db;
        $res = $db->update_profile($id, $login, $passwd, $name, $email, $disabled);
```

```
        if ($this->profile['id'] == $res)
        {
            $this->profile['login'] = $login;
            $this->profile['name'] = $name;
            $this->profile['email'] = $email;
            $this->set_session();
        }

        return $res;
    }

    //Создание профиля
    function create_profile($login, $passwd, $name, $email, $reg_date)
    {
        global $db;
        $res = $db->create_profile($login, $passwd, $name, $email, $reg_date);

        if ($res > 0)
        {
            $this->authorize($login, $passwd);
            $this->set_session();
        }

        return $res;
    }

    function is_auth() //Проверка, является ли пользователь авторизованным
    {
        return $this->authenticated;
    }

    function is_admin() //Проверка, является ли пользователь администратором
    {
        return $this->profile['is_admin'] == 't';
    }

    function get_profile() //Получение профиля пользователя
    {
        return $this->profile;
    }

    function has_rights($rights) //Получение прав пользователя
    {
        return ($this->profile['rights'][$rights] == 't') ? true : false;
    }
}

$user = new User();

//Чтобы понят принцип работы этих функций, необходимо ознакомится с файлами
```

```
sessions.php и db.php

?>
```

8.7 schema.sql

```
CREATE LANGUAGE plpgsql;

DROP TABLE IF EXISTS users;

CREATE TABLE users(
  id serial,
  login text,
  passwd text,
  name text,
  email text,
  role_id int DEFAULT 3 REFERENCES roles(id),
  reg_date date,
  last_login timestamp DEFAULT NULL,
  disabled bool DEFAULT FALSE,
  UNIQUE ( login ),
  UNIQUE ( email )
);

CREATE TABLE roles (
  id serial,
  name TEXT,
  profile_upd BOOL DEFAULT FALSE,
  users_upd BOOL DEFAULT FALSE,
  PRIMARY KEY (id)
);
```

8.8 mult.cgi

```
#include <stdio.h>
#include <stdlib.h>

int main( void )
{
  int x, y, i, j;
  char *query;

  /* Получене и расшифровка GET-запроса */
  query = getenv("QUERY_STRING");
  if (sscanf(query, "x=%d&y=%d", &x, &y) != 2)
   x = y = 1;

  if (x < 0)
   x *= -1;
  if (y < 0)
```

```
   y *= -1;

  /* Обязательная передача заголовка */
  printf("Content-Type: text/html;\n\n");

  /* Вывод HTML кода */
  printf("<html><head><title>Test</title><meta charset='utf-8' /></head><body>"
   "<form><label for='x'>X:</label> <input type='text' name='x' value='%d' /> "
   "<label for='x'>Y:</label> <input type='text' name='y' value='%d' /> "
   "<input type='submit' value='Считать' /></form><table>", x, y);

  for (j = 1; j <= y; j++)
  {
   printf("<tr>");
   for (i = 1; i <= x; i++)
     printf("<td>%d</td>", i * j);
   printf("</tr>");
  }

  printf("</table></body></html>");

  return 1;
}
```

8.9 all.css

```
/* Создание ключевых кадров анимации */
@keyframes load
{
    from
    {
        opacity: 0;
        transform: scale(0.9);
    }
    to
    {
        opacity: 1;
        transform: scale(1);
    }
}

/* Для Chome и Opera */
@-webkit-keyframes load
{
    from
    {
        opacity: 0;
        -webkit-transform: scale(0.9);
    }
    to
```

```
    {
        opacity: 1;
        -webkit-transform: scale(1);
    }
}

/* Для IE10 */
@-ms-keyframes load
{
    from
    {
        opacity: 0;
        -ms-transform: scale(0.9);
    }
    to
    {
        opacity: 1;
        -ms-transform: scale(1);
    }
}

/* Для Mozilla */
@-moz-keyframes load
{
    from
    {
        opacity: 0;
        -moz-transform: scale(0.95);
    }
    to
    {
        opacity: 1;
        -moz-transform: scale(1);
    }
}

*****************************************

#wrapper
{
    ...
    /* Запуск анимации */
    animation: load 0.6s linear;
    -webkit-animation: load 0.3s linear;
    -moz-animation: load 0.3s linear;
}
```

8.10 Скриншоты

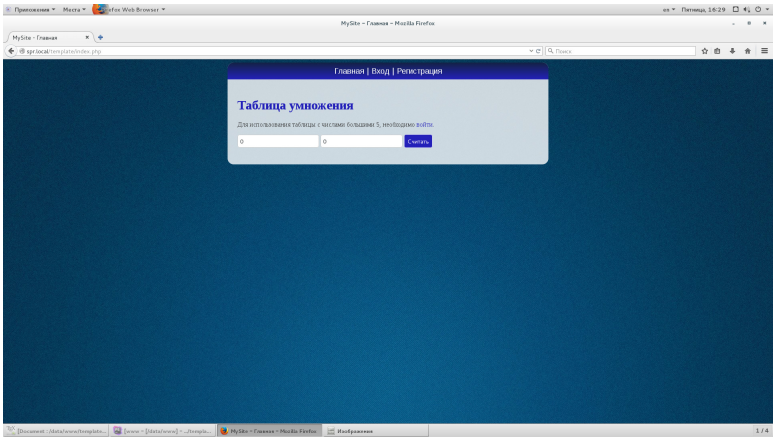

Рис. 1: Главная страница

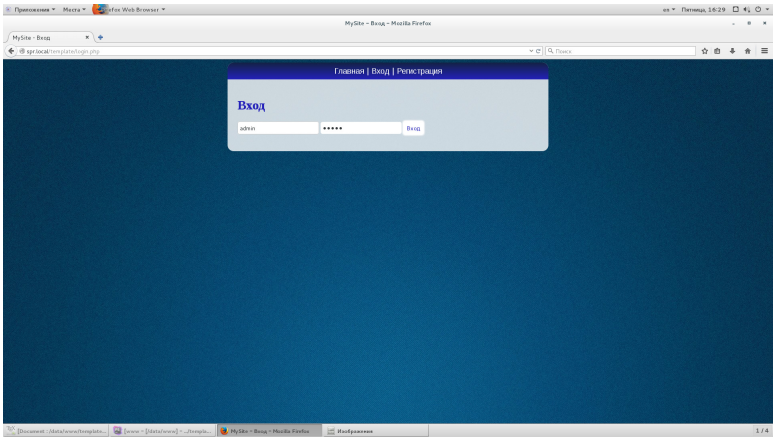

Рис. 2: Страница входа

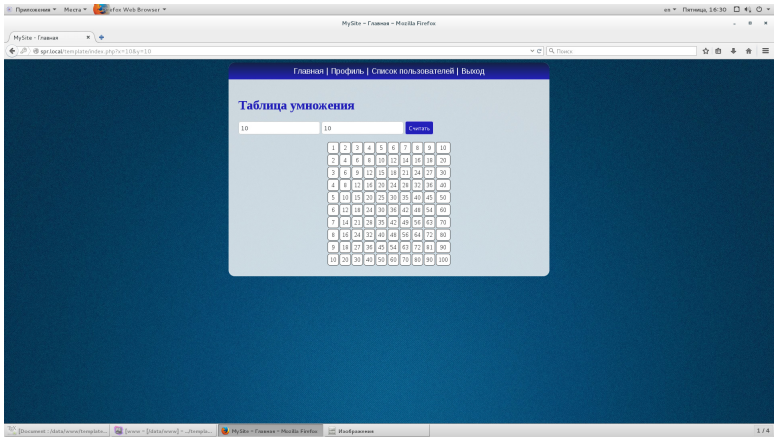

Рис. 3: Таблица умножения

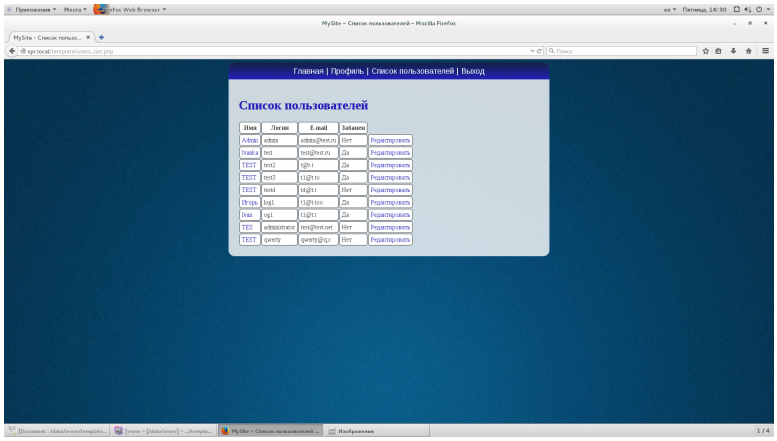

Рис. 4: Страница списка пользователей

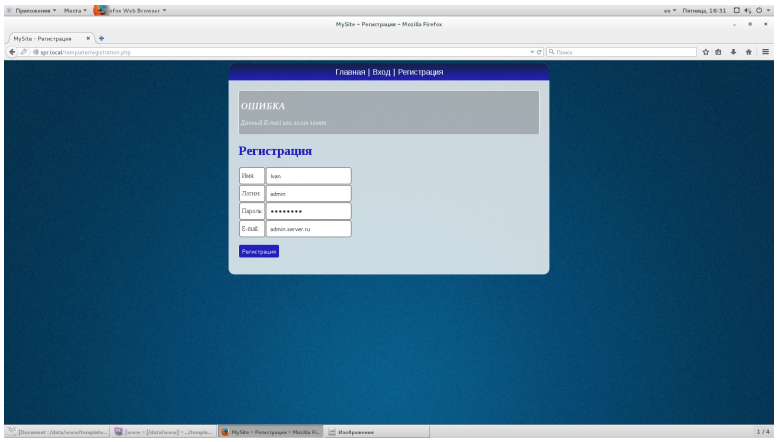

Рис. 5: Странице регистрации