Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа №1,2**

**Написание приложения для социальных сетей**

по дисциплине: «Сети и телекоммуникации».

Выполнил:

студент 4 курса, гр. ИВТАПбд-41

Кондратьев Павел Сергеевич.

Проверил:

преподаватель кафедры ВТ

Мартынов Антон Иванович.

г. Ульяновск, 2020

**Задание:**

Написание бота для социальной сети ВКонтакте, выводящем расписание пар в личные сообщения по запросу. В приложении реализован следующий функционал: подключение к HTML странице с расписанием пар (последующим сохранением стр. для минимизации повторного подключения к сайту), парсинг HTML верстки, заполнение базы данных SQL, содержащую id пользователя Вк, группу, url на расписание текущей группы, url картинки на 1 и 2 неделю расписания.

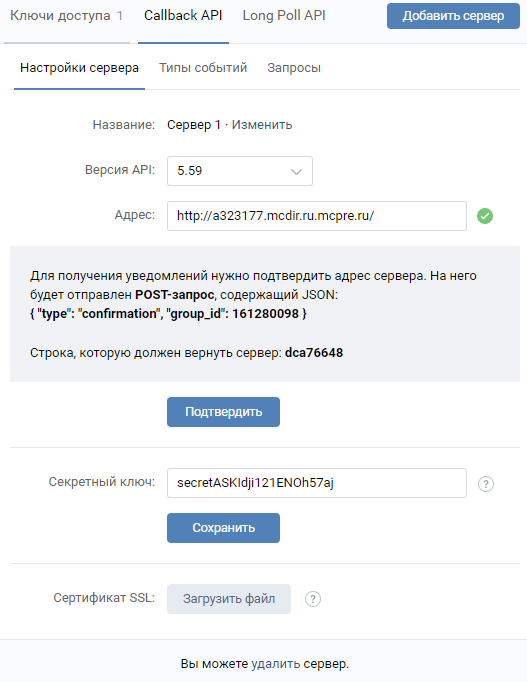
**Ход выполнения работы:**

Приложение должно использовать API ВКонтакте. API (application programming interface) — это посредник между разработчиком приложений и какой-либо средой, с которой это приложение должно взаимодействовать. API упрощает создание кода, поскольку предоставляет набор готовых классов, функций или структур для работы с имеющимися данными. API ВКонтакте — это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу. Вам не нужно знать в подробностях, как устроена база, из каких таблиц и полей каких типов она состоит — достаточно того, что API-запрос об этом «знает». Синтаксис запросов и тип возвращаемых ими данных строго определены на стороне самого сервиса.

Ответы на поступающие сообщения реализованы с помощью Callback API – инструмента для отслеживания активности пользователей в сообществе ВКонтакте.

Приложение было написано в ООП стиле, был реализован класс для работы с базой данных, класс, отвечающий за отправку сообщения пользователю, и класс-контейнер с методами обработки HTML страниц.

Для начала нужно подключить свой сервер в настройках сообщества («Управление сообществом» → «Настройки» → «Работа с API»). Выберите типы событий, данные о которых требуется получать, например, новые комментарии и новые фотографии.

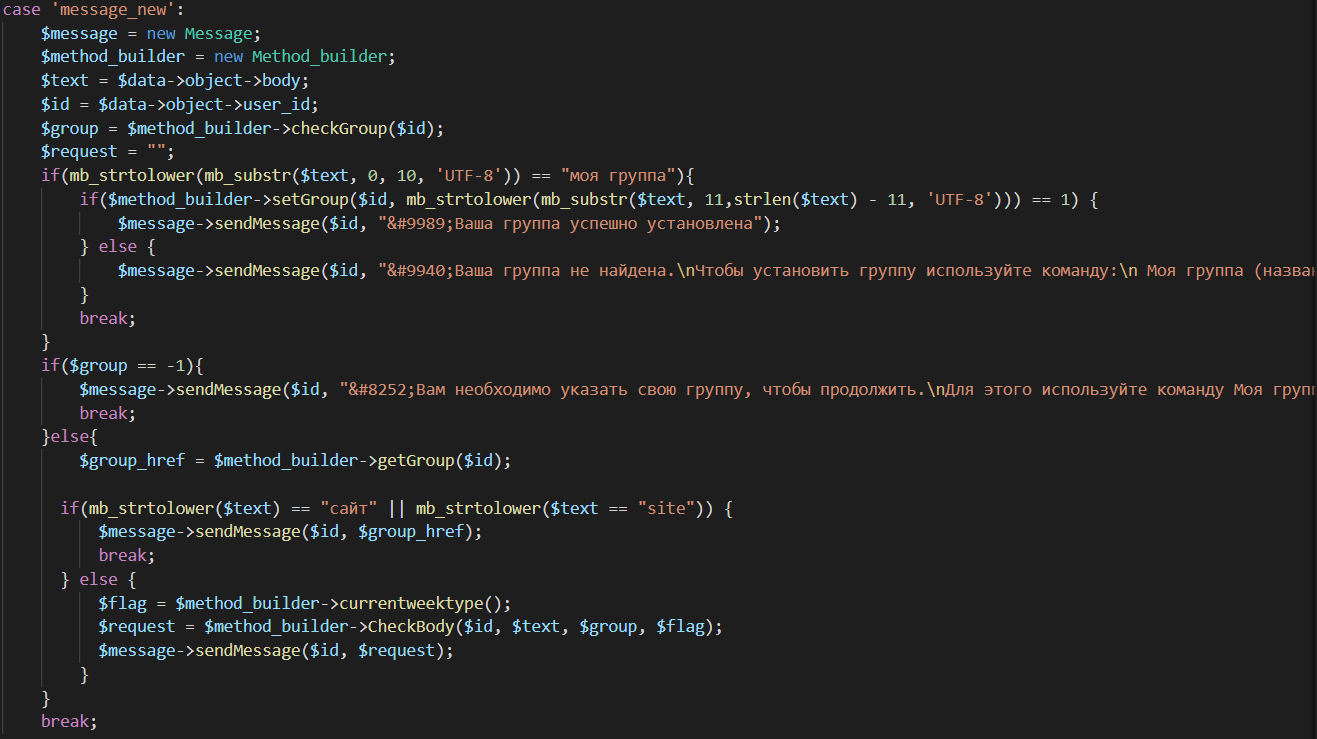


Когда в сообществе произойдет событие выбранного типа, ВКонтакте отправит на Ваш сервер запрос с данными в формате JSON с основной информацией об объекте, вызвавшем событие.

Реализованы следующие модули:

1. Класс message по отправке личных сообщений пользователю
2. Класс metod\_builder, со всеми возможными методами по работе с пользователем (сохранение, обновление данных о пользователе и его группы)
3. Класс query\_builder, по работе с запасай к бд

**Пример обработчика нового сообщения на сервере**



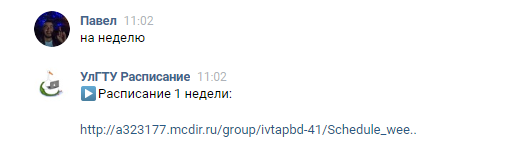
В событие message\_new, собирается информация о пользователе (его id, текст сообщения), после чего производиться проверка на наличие группы методом CheckGroup(), в зависимости от ответа выполняться следующие действия:

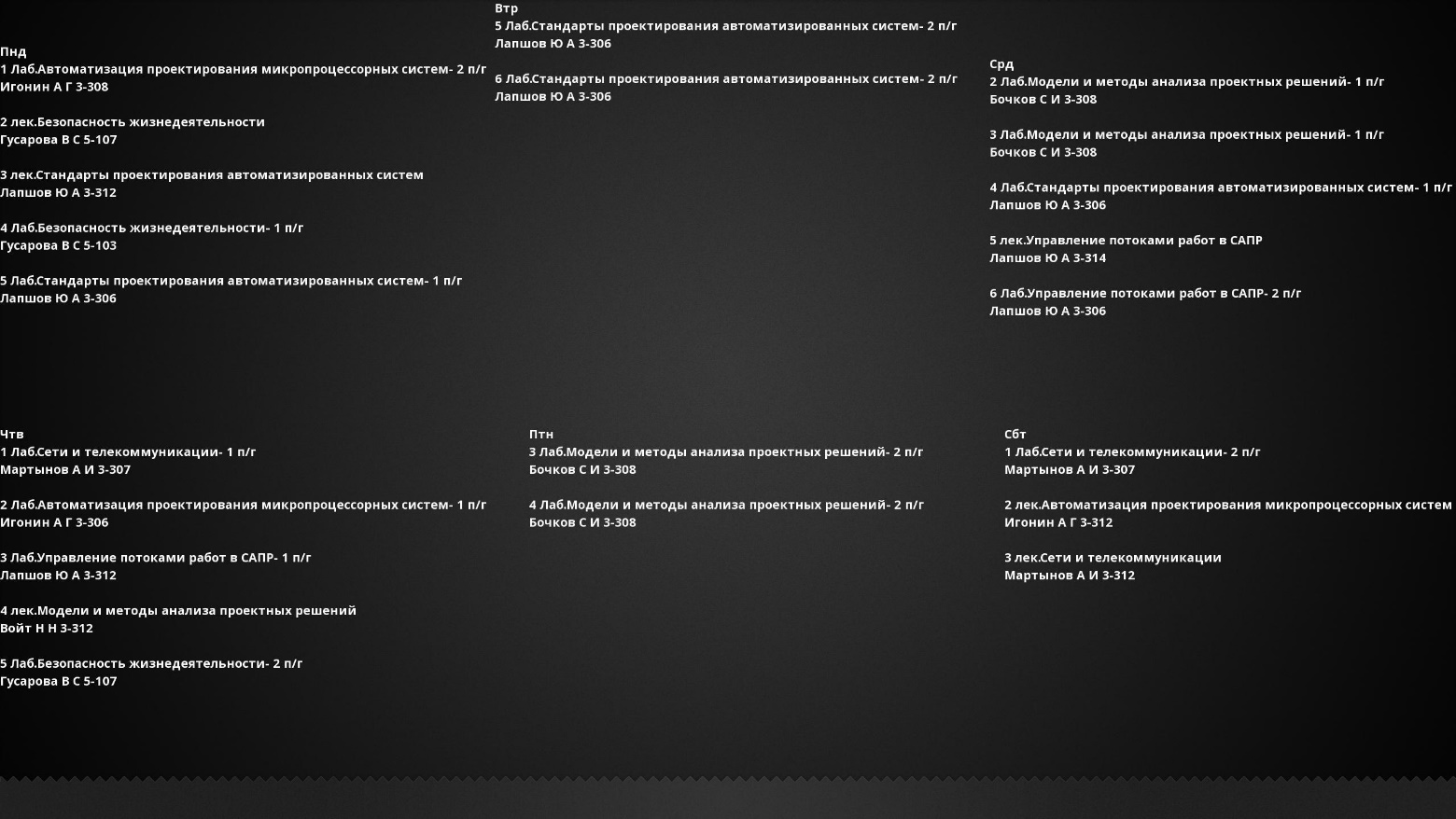
1. Если группы нету, то id пользователя сохраняется и посылается ответ с текстом об установлении текущей группы
2. Если группа есть то образуемся к бд и получаем ссылку на группу и в зависимости от текста сообщения выдаем ответ в методе CheckBody()
3. Производиться установка текущей группы и производиться сохранение html страницы группы, для единственного подключение к сайту

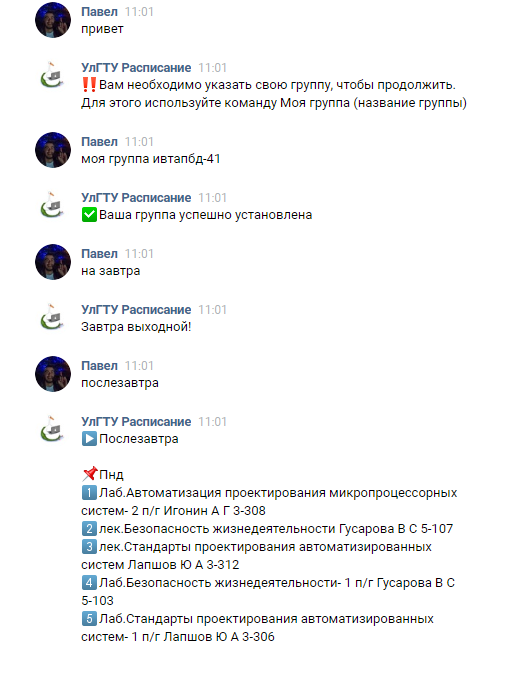
**Вывод:**

Данное приложение имеет огромный потенциал для массового использования и охватывает большую аудиторию. На сегодняшний день многие люди ежедневно пользуются социальными сетями и, следовательно, для них намного удобнее посмотреть расписание в контакте, а не на сайте.

**Пример работы:**







**Приложение 1**

**index.php**

<?php

require("method\_builder.php");

require("token.php");

require("message.php");

if(!isset($\_REQUEST)){

return;

}

$data = json\_decode(file\_get\_contents('php://input'));

switch ($data->type){

case 'confirmation':

echo VK\_CONFIRMATION\_CODE;

break;

case 'message\_new':

$message = new Message;

$method\_builder = new Method\_builder;

$text = $data->object->body;

$id = $data->object->user\_id;

$group = $method\_builder->checkGroup($id);

$request = "";

if(mb\_strtolower(mb\_substr($text, 0, 10, 'UTF-8')) == "моя группа"){

if($method\_builder->setGroup($id, mb\_strtolower(mb\_substr($text, 11,strlen($text) - 11, 'UTF-8'))) == 1) {

$message->sendMessage($id, "&#9989;Ваша группа успешно установлена");

} else {

$message->sendMessage($id, "&#9940;Ваша группа не найдена.\nЧтобы установить группу используйте команду:\n Моя группа (название группы)");

}

break;

}

if($group == -1){

$message->sendMessage($id, "&#8252;Вам необходимо указать свою группу, чтобы продолжить.\nДля этого используйте команду Моя группа (название группы)");

break;

}else{

$group\_href = $method\_builder->getGroup($id);

if(mb\_strtolower($text) == "сайт" || mb\_strtolower($text == "site")) {

$message->sendMessage($id, $group\_href);

break;

} else {

$flag = $method\_builder->currentweektype();

$request = $method\_builder->CheckBody($id, $text, $group, $flag);

$message->sendMessage($id, $request);

}

}

break;

}

?>

**method\_builder.php**

<?php

require("simple\_html\_dom.php");

require("constants.php");

require("query\_builder.php");

require("LImageHandler.php");

Class Method\_builder{

public function extract\_true\_tr\_from\_garbage($s) {

$tr\_starting\_pos = strpos($s, "<tr>");

$tr\_closing\_pos = strpos($s, "</TR>");

return substr($s, $tr\_starting\_pos + 4, $tr\_closing\_pos - $tr\_starting\_pos - 5);

}

public function today() {

$today = date('l');

return $this->convertDayEng($today);

}

public function divideStr($str){

return str\_replace("td", "td>\n<td", $str);

}

public function removeHTMLTagsWithNoContent($htmlBlob) {

$pattern = "#<p>(\s|&nbsp;|</?\s?br\s?/?>)\*</?p>#";

if (preg\_match($pattern, $htmlBlob) == 1) {

$htmlBlob = preg\_replace($pattern, '', $htmlBlob);

return $this->removeHTMLTagsWithNoContent($htmlBlob);

} else {

return '';

}

}

public function currentweektype(){

$today = date\_create(date('Y-m-d'));

$studyBeginning = date\_create('2020-02-03'); // Дата начала учебы

$interval = date\_diff($today, $studyBeginning);

$days = $interval->days;

$days = intdiv($days, 7);

if ($days % 2 == 0){

return 1;

}

else{

return 2;

}

}

public function CheckBody($id, $text, $group, $flag){

switch (mb\_strtolower($text)){

case "на день":

case "на сегодня":

case "сегодня":

$day = $this->today();

if ($day == "Воскресенье"){

$answer = "&#9654;Совет дня\n\nСегодня выходной!)";

} else {

$answer = "&#9654;Сейчас ".$flag." неделя\n\n".$this->printSchedule\_day($day, $group, $flag);

}

return $answer;

case "на завтра":

case "завтра":

$day = strtotime("+1 day");

if ($this->convertDayEng(date('l',$day)) == "Воскресенье"){

return "Завтра выходной!";

}

return "&#9654;Завтра\n\n".$this->printSchedule\_day($this->convertDayEng(date('l',$day)), $group, $flag);

case "на послезавтра":

case "послезавтра":

$day = strtotime("+2 day");

if ($this->convertDayEng(date('l',$day)) == "Воскресенье"){

return "Послезавтра выходной!";

}

return "&#9654;Послезавтра\n\n".$this->printSchedule\_day($this->convertDayEng(date('l',$day)), $group, $flag);

case "на неделю":

case "неделя":

return "&#9654;Расписание " . $flag . " недели:\n\n".$this->printSchedule\_week($id, $group, $flag);

case "на 1 неделю":

case "1 неделя":

case "1":

return "&#9654;Расписание 1 недели:\n\n".$this->printSchedule\_week($id, $group, 1);

case "на 2 неделю":

case "2 неделя":

case "2":

return "&#9654;Расписание 2 недели:\n\n".$this->printSchedule\_week($id, $group, 2);

case "help":

case "помощь":

case "команды":

return "&#128204; Текущие команды:\n\n1&#8419;Моя группа Название группы\n\nУстанавливает текущую группу\n\n2&#8419;на день/на сегодня/сегодня\n\nВозвращает пары на текущий день\n\n3&#8419;на завтра/завтра\n\nВозвращает пары на завтра\n\n4&#8419;на послезавтра/послезавтра\n\nВозвразает пары на послезавтра\n\n5&#8419;на неделю/неделя\n\nВозвразает пары на текущую неделю\n\n6&#8419;на 1 неделю/1 неделя/1\n\nВозвразает пары на первую неделю\n\n7&#8419;на 2 неделю/2 неделя/2\n\nВозвразает пары на вторую неделю";

default:

return "&#9999;Что я могу\n1&#8419;Узнай команды написав: help/помощь/команды\n2&#8419;Возвращать пары на сегодня/завтра/послезавтра\n3&#8419;Возвращать пары на неделю/на 1 неделю/на 2 неделю\n4&#8419;Моя группа (Название группы)\n5&#8419;Возвращать сайт\n6&#8419;&#128293;В разработке&#128293;";

}

}

public function getGroup($id){

$query\_builder = new Query\_builder();

$result = $query\_builder->select("user", 'href\_group', 'user\_id = "' . $id . '" LIMIT 1');

if($result){

return $result[0][0];

}else{

return -1;

}

}

public function setGroup($id, $group){

$group\_href = "";

$ch = curl\_init('https://www.ulstu.ru/schedule/students/part1/raspisan.htm');

// Указываем, что результат должен записаться в переменную

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

// Выполняем CURL запрос

$data = curl\_exec($ch);

// Передаём данные, полученные с помощью CURL в SimpleHtmlDom функцию

$html = str\_get\_html($data);

foreach($html->find('a') as $element){

$tag\_text = strip\_tags($element->innertext);

if (mb\_strtolower($tag\_text) == $group) {

$group\_href = $element->href;

break;

}

}

if(!$group\_href) {

return -1;

}

$path\_schedule = 'https://www.ulstu.ru/schedule/students/part1/' . $group\_href;

$query\_builder = new Query\_builder();

$array = array(

'user\_group' => $this->translit($group),

'href\_group' => $path\_schedule

);

$query\_builder->update("user", $array, "user\_id = " .$id);

$html->clear();

unset($html);

$ch = curl\_init($path\_schedule);

// Указываем, что результат должен записаться в переменную

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

// Выполняем CURL запрос

$file\_schedule = str\_get\_html(curl\_exec($ch));

$file\_schedule = str\_replace("charset=windows-1251", "charset=utf-8", $file\_schedule);

mkdir("group/".$this->translit($group));

file\_put\_contents('./group/'.$this->translit($group).'/schedule.html', $file\_schedule);

return 1;

}

public function checkGroup($id){

$query\_builder = new Query\_builder();

$array = array(

'user\_id' => $id,

'user\_group' => ""

);

$result = $query\_builder->select("user", null, 'user\_id = "' . $id . '" LIMIT 1');

if($result){

$res = $query\_builder->select("user", "user\_id", 'user\_group = "' . $result[0][1] . '" LIMIT 1');

$week = $query\_builder->select("user", "first\_week" . ",second\_week", 'user\_id = "' . $res[0][0] . '" LIMIT 1');

$array = array(

'first\_week' => $week[0][0],

'second\_week' => $week[0][1]

);

$query\_builder->update("user", $array, "user\_id = " .$id);

return $result[0][1];

}else{

$query\_builder->insert("user", $array);

return -1;

}

}

public function entityIdentificationPair($pair) {

$result = "";

$search = 0;

foreach(str\_get\_html($pair)->find('p') as $item) {

if($search >= 1) {

if($item->innertext && trim($item->innertext) != '\_') {

$item->innertext = "\n" . $search . '&#8419;' . $item->innertext;

} else {

$item->innertext = '';

$item = $this->removeHTMLTagsWithNoContent($item);

}

$result .= $item;

$search++;

} else {

$result .= "&#128204;" . $item;

$search++;

}

}

return $result;

}

public function printSchedule\_day($day, $group, $flag){

$target = file\_get\_contents('./group/'.$group.'/schedule.html');

$table = str\_get\_html($target);

foreach($table->find('tr') as $element){

$table\_day = strip\_tags($element->find('td p', 0));

if ($day === $table\_day) {

$pair = $this->extract\_true\_tr\_from\_garbage($element);

if ($flag == 1) {

$week .= $this->entityIdentificationPair($pair);

$table->clear();

unset($table);

return strip\_tags($week);

break;

} else {

$flag--;

}

}

}

return 0;

}

public function draw\_week($table, $start, $finish, $name, $group) {

$x = 0;

$y = 0;

// Создаем экземпляр класса LImageHandler

$ih = new LImageHandler;

// // Подключаем выбранный шрифт текста

$fontPath = './OpenSans-Bold.ttf';

// Путь к оригинальному изображению

$imagePath = './template.jpg';

// Указываем размер шрифта

$fontSize = 16;

// Задаем цвет

$colorArray = array(255, 255, 255);

$position = array(

0 => LImageHandler::CORNER\_LEFT\_TOP,

1 => LImageHandler::CORNER\_CENTER\_TOP,

2 => LImageHandler::CORNER\_RIGHT\_TOP,

3 => LImageHandler::CORNER\_LEFT\_BOTTOM,

4 => LImageHandler::CORNER\_CENTER\_BOTTOM,

5 => LImageHandler::CORNER\_RIGHT\_BOTTOM,

);

$i = 0;

$passFirstRow = 0;

foreach($table->find('tr') as $element) {

if($passFirstRow >= $start && $passFirstRow <= $finish) {

$pair = $this->extract\_true\_tr\_from\_garbage($element);

$week = "\n" . str\_replace("<br>", "\n", $this->entityIdentificationPair\_def($pair));

// Загружаем изображение

if($passFirstRow == $start) {

$imgObj = $ih->load($imagePath);

$y -= ceil(strlen($week) / 2);

} else {

$imgObj = $ih->load("./group/" . $group . "/" . $name . ".jpg");

if($i == 1) {

$y = -ceil(strlen($week) / 2);

}

if($i == 2) {

$y = -ceil(strlen($week) / 2);

}

}

if($passFirstRow == ceil(($start + $finish) / 2)) {

$y = 0;

$y += 700;

}

$imgObj->text(strip\_tags($week), $fontPath, $fontSize, $colorArray, $position[$i], $x, $y);

$i++;

$imgObj->save("./group/" . $group . "/" . $name . ".jpg");

}

$passFirstRow++;

}

$url = "http://a323177.mcdir.ru/group/" . $group . "/" . $name . ".jpg";

return $url;

}

public function printSchedule\_week($id, $group, $flag) {

$target = file\_get\_contents('./group/'.$group.'/schedule.html');

$table = str\_get\_html($target);

$query\_builder = new Query\_builder();

if($flag == 1) {

$start = 2;

$finish = 7;

$result = $query\_builder->select("user", 'first\_week', 'user\_id = "' . $id . '" LIMIT 1');

}

if($flag == 2) {

$start = 11;

$finish = 16;

$result = $query\_builder->select("user", 'second\_week', 'user\_id = "' . $id . '" LIMIT 1');

}

if(empty($result)){

return $result[0][0];

}else {

$url = $this->draw\_week($table, $start, $finish, "Schedule\_week-" . $flag, $group);

if($flag == 1) {

$array = array(

'first\_week' => $url

);

} else {

$array = array(

'second\_week' => $url

);

}

$query\_builder->update("user", $array, "user\_id = " .$id);

return $url;

}

}

}

?>