DOI 10.37882/2223–2966.2022.10.26

АВТОМАТИЗИРОВАННОЕ НАПОЛНЕНИЕ МАТЕРИАЛАМИ СООБЩЕСТВ «ВКОНТАКТЕ»

Похорукова Мария Юрьевна
Технический институт (филиал) Северо-Восточный федеральный университет имени М.К. Аммосова Г. Нерюнгри
maria.pokhorukova@gmail.com
Пиманов Андрей Евгеньевич
Технический институт (филиал) Северо-Восточный федеральный университет имени М.К. Аммосова Г. Нерюнгри

AUTOMATED FILLING OF VKONTAKTE COMMUNITIES WITH MATERIALS

M. Pohorukova
A. Pimanov

Summary. The transfer of some everyday work processes to automated mode is an effective way to facilitate routine work, as well as reduce the time for its implementation.

In this article, an algorithm is considered, the purpose of which is to automatically transfer the process of filling communities «In Contact» with various materials. In the course of the work, a general–purpose scripting language, PHP, was used through its interaction with the VK API. Analytical work has been carried out to identify the disadvantages and advantages of using this algorithm. The article contains a detailed description of the implementation of the algorithm, taking into account the optional costs of ensuring its operability.

The result of the work is to obtain a script that allows you to automate the process of filling communities with materials, according to specified criteria, from sources specified by the user.

This script can be used to optimize some processes and modified in any convenient way, taking into account all the needs necessary for its implementation.

Keywords: algorithm, filling with materials, communities «VKontakte», process automation, publishing posts, script, VK API, PHP.

Аннотация. Перевод некоторых повседневных рабочих процессов в автоматизированный режим, является эффективным способом облегчить рутинную работу, а также сократить время на её выполнение.

В данной статье рассмотрен алгоритм, цель которого — перевод в автоматический режим процесса наполнения сообществ «В Контакте» различными материалами. В ходе работы использован скриптовый язык общего назначения — PHP, посредством его взаимодействия с VK API. Проведена аналитическая работа по выявлению недостатков и преимуществ использования данного алгоритма. Статья содержит подробное описание реализации алгоритма с учетом необязательных затрат на обеспечение его работоспособности.

Результатом работы является получение скрипта, позволяющего автоматизировать процесс наполнения сообществ материалами, по заданным критериям, из указанных пользователем источников.

Данный скрипт может быть использован для оптимизации некоторых процессов и видоизменен любым удобным способом, с учетом всех потребностей, необходимых для его реализации.

Ключевые слова: алгоритм, наполнение материалами, сообщества «В Контакте», автоматизация процессов, публикация постов, скрипт, VK API, PHP.

Для того чтобы действительно понимать пользу алгоритма, описанного в данной статье, необходимо иметь опыт работы с сообществами «В Контакте» или понимать, что они из себя представляют. Цель алгоритма, о котором пойдёт речь, заключается в наполнении сообществ материалами, посредством их автоматизированного заимствования из избранных источников, с последующей публикацией.

В настоящее время возможность автоматизированной наполняемости сообществ материалами, является очень актуальной. Следует отметить, что данный алгоритм ориентирован на сообщества, для которых оригинальность используемого материала не имеет ключевого значения.

Применение данного алгоритма позволяет владельцу сообщества не заниматься самостоятельным подбором материалов и снижает трудозатраты, связанные с поддержанием одного или нескольких критериев рейтинга сообщества. Из обременительных тягот использования алгоритма можно назвать необходимость формирования списка источников, из которых будут заимствоваться материалы, регулярная проверка качества заимствованных материалов и самое обременительное из вышеперечисленного — обеспечение

Таблица 1. Конфигурация виртуального сервера

Процессор	AMD EPYC7742 (2 ядра)
Оперативная память	4 Гб
Скорость соединения	До 100 Мбит/сек
Виртуализация	KVM
Накопитель	NVME40 Гб
Операционная система	Ubuntu 20.04
Apache	2.4.41
MySQL	8.0.26
ISPmanager Lite	6.35.1

инфраструктуры для работы алгоритма. Оценивая все плюсы и минусы, можно с уверенностью сделать вывод, что существенность плюсов значительно перевешивает несущественность перечисленных минусов.

Перед началом работы, владельцу сообщества необходимо иметь четкие представления о том, какие трудозатраты ему необходимо перевести в автоматизированный режим. Рассмотрим один из возможных вариантов. Пусть это будет некий скрипт, расположенный на виртуальном сервере, выполняющий свою работу по заданным ему правилам. Публикация постов будет происходить один раз в час. Данный алгоритм должен работать со списком источников, которые ему заданы, и не использовать материалы, в которых используются слова, указывающие на наличие ссылок на другие источники.

Для взаимодействия с социальной сетью «ВКонтакте» воспользуемся VK API [1]. API «ВКонтакте» — это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу. Подробнее об этом можно узнать на сайте «В Контакте» в разделе для разработчиков.

Далее следует определится с удалённым сервером. В нашем случае это будет виртуальный сервер, взятый в аренду со следующей конфигурацией (таблица 1).

Стоить отметить, что для поставленной задачи, выбранная конфигурация является избыточной. Как правило, если вы возьмёте виртуальный сервер в аренду, на нём будет предустановлен весь софт, с которым нам придётся столкнуться.

Для взаимодействия с VK API из-под удалённого сервера, хорошо подойдёт язык PHP [2]. PHP — Скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. Используемая нами версия — 7.4.3.

На следующем этапе сформируем источник с данными используя среду phpMyAdmin. Создаём базу [3] данных с таблицей из четырёх столбцов: ID, group_id, name и used. Для полей ID и group_id определяем целочисленный тип, и для ID соответственно задаём первичный ключ. Поле name несёт вспомогательный функционал и используется для хранения названия источника. Для used используем tinyint длины 1, и определённый по умолчанию как 0. Поле group_id содержит идентификаторы сообществ, чьи материалы будут использоваться, и оно является отрицательным числом. Добавим некоторое количество записей.

Сгенерируем файл с правилами работы алгоритма. В нём мы будем хранить токен пользователя для авторизации в сообществе, используемую версию VK API, идентификатор пользователя и сообщества, а также массив запрещённых слов, которые не должны встретиться в посте, из которого будет заимствована информация.

Начнём наполнять config.php. Для этого создадим сообщество «В Контакте». Последнюю версию API можно узнать в документации в разделе для разработчиков на сайте «ВКонтакте», в данном случае мы будем использовать 5.131. Токен пользователя можно получить из приложений «ВКонтакте», во время передачи разрешений. Идентификаторы сообщества и пользователя можно получить на странице сообщества и пользователя соответственно. В массив запрещённых слов войдут фрагменты текста, указывающие на наличие в тексте ссылок. Таким образом, конфигурационный файл будет выглядеть следующим образом:

Листинг 1

```
<?php

// Общие параметры
const VERSION = "5.131";

// Пользовательские параметры
const USER_TOKEN = "…";
const USER_ID = 347771519;

// Идентификатор сообщества
const GROUP_ID = –209936982;

// Параметры исполнения скриптов
const FORBIDDEN_WORDS_IN_THE_POST =
array('http', 'www', 'club', 'id');
```

Создадим файл vk.php в котором будем хранить класс с публичными функциями [4], к которым мы будем обращаться для взаимодействия с VK API. Сначала подтянем конфиги записав их в переменные. Добавим функцию call_with_user_token выполняющую запросы к VK API. Данная функция будет принимать название метода и массив данных для формирования JSON запроса. При исполнении функции в полученный массив добавится информация об используемой версии VK API и токен подтверждающий пользователя, отправившего запрос. Формируем адрес из имеющегося url "https://api.vk.com/method/", названия метода и массива с параметрами, отправляемыми на сервер. Отправляем запрос с помощью curl. Ответ от VK функция вернёт для дальнейшей работы с ним.

Листинг 2

```
public function call_with_user_token($method,
$params = []) {
$params['access_token'] = $this->user_token;
$params['v'] = $this->version;
$url = 'https://api.vk.com/method/'.$method.'?'.http_
build_query($params);
$curl = curl_init($url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
$json = curl_exec($curl);
curl_close($curl);
return json_decode($json);
}
```

Остальные функции этого класса в большинстве случаев будут обращаться к данной функции. Например, метод для сохранения изображений на стену сообщества будет выглядеть так:

Листинг 3

```
public function photos_save_wall_photo($photo,
$server, $hash) {
return $this->call_with_user_token('photos.
saveWallPhoto', [
'group_id' => $this->group_id * –1,
'photo' => $photo,
'server' => $server,
'hash' => $hash
]);
}
```

Также в этом классе предусмотрена функция для загрузки изображений на сервера ВКонтакте — photos_upload_server. Работает аналогично методу call_with_user_token.

Листинг 4

```
public function photos_upload_server($url, $image_
path) {
$params['photo'] = new CURLFile($image_path);
$params['access_token'] = $this->user_token;
$params['v'] = $this->version;
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL, $url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'POST');
curl_setopt($curl, CURLOPT_POST, true);
curl_setopt($curl, CURLOPT_POSTFIELDS, $params);
$json = curl_exec($curl);
curl_close($curl);
return json_decode($json);
}
```

Закончим с vk.php данным набором функций и приступим к основному скрипту, которому дадим название publish_post.php. Сначала подтягиваем vk.php с помощью require_once.

Листинг 5

```
require_once(__DIR__.'vk.php');
$vk = new VK();
```

Таким же способом подтягиваем config.php и две переменные из него: массив с запрещёнными словами и идентификатор владельца страницы, на чью стену будут загружаться изображения, а после публиковаться на странице сообщества.

Листинг 6

```
require_once(__DIR__.'config.php');
$forbidden_words_in_the_post = FORBIDDEN_
WORDS_IN_THE_POST;
```

```
$owner_id = USER_ID;
```

Далее создаём подключение к базе данных:

Листинг 7

```
$mysqli = new mysqli('localhost', 'user', 'password_for_user', 'set_source');
```

Затем функцию получающую случайную запись из базы данных, в которых поле "used" равно 0. Для работы с базой данных используем язык SQL [5].

Листинг 8

```
function get_group_id_source($mysqli)
{
$group_id_source = $mysqli->query("SELECT `group_id` FROM `set_source`.`source` WHERE `used` = '0' ORDER BY RAND() LIMIT 1");
return $group_id_source->fetch_row();
}
```

Далее получаем текущее время, а точнее час. В переменную записываем результат выполнения функции. Проверяем начались ли новые сутки или закончились ли записи из столбца "used" со значением 0. Если выполняется хоть одно из условий, присваиваем всем записям 0 в поле "used" и обновляем значение переменной со значением функции.

Далее мы воспользуемся циклом while с проверкой на существование записи с полем "used" равным единице. Пока такая запись есть — цикл будет выполнятся. На этом этапе в нашей переменной $group_id_source хранится идентификатор сообщества, из которого мы будем заимствовать материал. Начнём исполнение цикла вот с такой строчки кода:

Листинг 9

```
$mysqli->query("UPDATE `set_source`.`source` SET `used` = '1' WHERE `group_id` = '$group_id_source[0]';");
```

Она меняет поле "used" в записи с выбранным сообществом на 1. Таким образом, мы запоминаем из каких сообществ мы уже брали материалы в ближайшее время. Чтобы получить материал из сообщества мы заведём в vk.php функцию, обращающуюся к методу wall.get.

Листинг 10

```
public function wall_get($owner_id, $count, $extended) {
return $this->call_with_user_token('wall.get', [
'owner_id' => $owner_id,
'count' => $count,
'extended' => $extended
]);
}
```

Данная функция вернёт JSON полученный от «В Контакте», с ним мы и продолжим работу. Подробную информацию о каждом методе, который мы будем использовать можно найти в документации VK API. В owner_id указываем идентификатор сообщества, в count указываем значение 2, означающее что мы хотим получать 2 записи со стены сообщества. Нам нужна информация о второй записи на стене, так как вероятно, что первая запись на стене произвольного сообщества является закреплённой записью и может содержать справочную информацию.

Таким образом, мы получаем информацию о записи из сообщества:

Листинг 11

```
$wall_get = $vk->wall_get($group_id_source[0], 2, 1);
```

Вытягиваем количество фотографий и текст из записи:

Листинг 12

```
$attachments_length = count($wall_get->response->items[1]->attachments);
$text = $wall_get->response->items[1]->text;
```

Теперь проверим наш текст на запрещённые фрагменты. Если такие найдутся мы присвоим переменной $text значение check.

Листинг 13

```
for ($i = 0; $i < count($forbidden_words_in_the_post); $i++)
{
if (strpos($text, $forbidden_words_in_the_post[$i])!== false)
{
$text = 'check';
break;
}
}
```

Таким образом мы выполняем проверку на наличие изображений в посте, наличие запрещённых слов, а также проверяем помечен ли пост как рекламный и есть ли у него указанные источники.

Листинг 14

```
if ($attachments_length and $text!='check' and!$wall_
get->response->items[1]->marked_as_ads and!$wall_
get->response->items[1]->copyright->link)
```

Если условия выполняются, мы удаляем все изображения из папки image (предварительно её нужно создать).

Листинг 15

```
array_map('unlink', glob(__DIR__.'/image/*'));
```

Теперь мы должны сохранить все изображения из поста и загрузить на сервер «В Контакте», а также после загрузки получить идентификаторы вложений. Делается это с помощью двух методов: photos.getWallUploadServer — получаем ссылку для загрузки изображения, photos.saveWallPhoto — сохраняем изображение, возвращаем его идентификатор. После получения ссылки для загрузки мы воспользуемся функцией из vk.php — photos_upload_server. Сохранив изображение с помощью последнего метода, получим JSON, данные из которого нам помогут получить идентификатор вложения.

Листинг 16

```
$attachments = '';
for ($i = 0; $i < $attachments_length; ++$i)
if ($wall_get->response->items[1]->attachments[$i]-
>type == 'photo')
{
$image_file = file_get_contents($wall_get->response-
>items[1]->attachments[$i]->photo->sizes[count($wall_
get->response->items[1]->attachments[$i]->photo-
>sizes) — 1]->url);
file_put_contents(__DIR__.'/image/'. $i. '.jpg', $image_
file);
$photos_get_wall_upload_server = $vk->photos_get_
wall_upload_server();
$photos_upload_server = $vk->photos_upload_
server($photos_get_wall_upload_server->response-
>upload_url, __DIR__.'/image/'. $i. '.jpg');
$photos_save_wall_photo = $vk->photos_save_wall_
photo($photos_upload_server->photo, $photos_upload_
server->server, $photos_upload_server->hash);
$attachments = $attachments. 'photo'. $owner_id. '_'.
$photos_save_wall_photo->response[0]->id. ',';
}
```

В переменной $attachments сохранены все идентификаторы вложений, разделённых через запятую.

Функция обращающаяся к методу photos.getWallUploadServer передаёт VK API идентификатор сообщества.

Функция загружающая фотографии на сервер ВКонтакте передаёт лишь изображение.

Функция обращающаяся к методу photos.saveWallPhoto передаёт VK API параметры полученные от предыдущей функции.

Для публикации записи на стене сообщества заведём функцию в vk.php:

Листинг 17

```
public function wall_post($message, $attachments,
$publish_date) {

return $this->call_with_user_token('wall.post', [
'owner_id' => $this->group_id,
'message' => $message,
'attachments' => $attachments,
'publish_date' => $publish_date
]);
}
```

Параметр publish_date принимает время, в которое нужно опубликовать запись в Unix формате. Время публикации записи в нашем случае будет определяться тем моментом, когда начал выполняться скрипт, плюс сутки.

Листинг 18

```
$vk->wall_post('', $attachments, date(U) + 86400);
break;
```

Если запись будет опубликована, тогда программа наткнётся на break и выйдет из цикла while. Если запись не будет опубликована, то значение переменной $group_id_source обновится. Прочитать весь код можно на GitHub [6]. Чтобы скрипт исполнялся каждый час, добавим его в планировщик cron через ISPmanager. Следует отметить, что данный скрипт можно изменить так, как нам будет это удобно. К примеру, можно добавить рассылку в личные сообщения о новых записях. Данному скрипту можно дать и другие реализации. В том числе реализацию этого скрипта, которая будет срабатывать раз в сутки и заготавливать записи на весь день вперёд. Следует учесть одну важную деталь данного алгоритма, чтобы публикуемые записи были достаточно разнообразными, необходима база данных из сообществ, которая по своему объёму будет превышать количество публикуемых за сутки постов желательно в 2 раза.

ЛИТЕРАТУРА

1. Официальный веб-сайт «ВКонтакте», раздел «Для разработчиков» [Электронный ресурс]. URL: https://dev.vk.com/reference (дата обращения: 04.02.2022).
2. Официальный веб-сайт «php.net», раздел «Руководство по PHP» [Электронный ресурс]. URL: https://www.php.net/manual/ru/index.php (дата обращения: 04.02.2022).
3. Карпова И.П. Базы данных. — М.: Питер, 2013. С. 3–31
4. Кузнецов М.В., Симдянов И.В. Самоучитель PHP 7. — СПб.: БХВ-Петербург, 2018. С. 143–151.
5. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL. Полное руководство. — Вильямс, 2018. С. 95–148.
6. Официальный веб-сайт «GitHub», репозиторий [Электронный ресурс]. URL: https://github.com/Aiciokizava/automated-filling-of-VKontakte-communities-with-materials

Журнал «Современная наука: актуальные проблемы теории и практики»

Северо-восточный федеральный университет им М.К. Аммосова