ECONTECHMOD. AN INTERNATIONAL QUARTERLY JOURNAL – 2017. Vol. 6. No. 2. 3–8

Use of a cloud storage for implementation informational processes

N. Boyko

Lviv Polytechnic National University, Lviv, Uktaine; e-mail: nataliya.i.boyko@lpnu.ua

Received February 15.2017: accepted June 2.2017

Abstract. In this paper, description of a concept of cloud storage is offered. One of the most practical methods of storing required information to cloud storage is also considered. Method of creating screenshots is proposed. Theoretical research is carried out and there are justified advantages and disadvantages of different methods of storing information in social networks. The best technique of creating screenshots has been practically implemented. Problems that might come up while working with approach given in this article and its solutions are stipulated. There is analyzed a necessity of setting a zoom parameter which is to transfer after transmitting size value of the picture in social networks. In the article the parameter that specifies the width of the final image and clearly affects the quality of the image is also considered. There is analyzed the effectiveness of creating an information system that saves time for such information processes as tracking the photo and its comments. In the paper the task of changing bets, which are not immediately fixed in social networks is optimized. Also there is implemented in practice a scalability problem of information processes in social networks. In the article a separation of the script is also put into practice. One part of which directly performs the request of image and downloads it to cloud storage. With help of another part the information process is transmitted on the photo where the user identifies.

Key words: information process, script, information system, cloud storage, screenshot, cloud service, application programming interface, software.

INTRODUCTION

Cloud storage is a model of data storage where the digital data is stored in logical pools, the physical storage includes multiple servers (and often in different locations), the physical environment is usually owned and managed by a hosting company. These cloud storage providers are responsible for keeping the data available and accessible, and for physical environment working. People buy storage capacity from the providers to store data [1, 6–9].

Cloud storage services can be available through a co-located cloud computer service, or a web service application programming interface (API) or by applications that uses the API [2, 19].

Cloud storage is based on highly virtualized infrastructure, also it is like broader cloud computing in the view of accessible interfaces, near-instant elasticity and scalability, multi-tenancy, and metered resources [4, 16].

Cloud storage typically refers to a hosted object storage service, but the term has broadened to include other types of data storage that are now available as a service, like block storage.

Cloud storage is [6, 18]:

- made up of many distributed resources, but acts as one, also known as federated or a cooperative storage cloud architecture;
- high fault tolerant through redundancy and distribution of data;
- high-strength through the creation of versioned copies.

This article presents the method of taking screenshots of a community in social network, called “Vkontakte”. This process was useful in managing project, called “Date auction” (the “Auction”). The purpose of creating this project was the need to notice all the comments (bets) in the community. Since, it is difficult to handle all the comments manually this method of automation was chosen [4, 7].

THE PRELIMINARY SEARCH, ANALYSIS OF THE PROBLEM

Problem of the statement: to automate the process of creating screenshots with bets.

According to the task, there are three key issues:

1. Find out whether the bet is placed.
2. Take a screenshot of the last bet (the word “last” will later become the key to this paragraph).
3. Save the screenshot to cloud storage (where and how).

THE FIRST PROBLEM: NOTIFICATION OF PLACING THE BET

Whatever, is it a new post on the wall or question in discussion or comment to the photo, social network “Vkontakte” provides an open feedback for any event in the community [14, 20]. All necessary documentation is freely available, the link (https://vk.com/dev/

callback_api). After reviewing the documentation, the next step is to get a community secret key, which is used in queries. Having received the key, the script must be configured, it will handle requests, which, in turn, will come with all necessary information in JSON format. Another advantage is the possibility of forming requests for receiving additional data based on the obtained information.

An example of this request:

{ "type": "photo_comment_new",
"object": {
*"id": 4**2,*
*"from_id": 15*****30,*
"date": 1478607535,
"text": "Hello)",
*"photo_owner_id": -46****81,*
*"photo_id": 43*****42 },*
*"group_id": 46****81}*

THE SECOND PROBLEM: CREATING SCREENSHOTS

It is necessary to consider a few solutions:

1. Use your server.
2. Seek the assistance of third-party services.

Advantages and disadvantages are obvious: while using your server, it is necessary to set up the software, in case of using a third-party service, solution consists in correct formation of the request. Since, the second option is simpler to implement, it was selected, namely a “Site-Shot” service. It is easy and convenient in usage, also all necessary documentation is freely available [8, 17]. This service consists of a free limited number of screenshots (https://www.s-shot.ru/doc) and paid service (for each picture https://www.s-shot.ru/doc_extended). Free version is enough for efficient testing, however, it is worth to buy paid version for usage in a consistent way. To take a webpage screenshot it is enough to form an URL, including all the required parameters.

For example, to take a screenshot of “Google” it is enough to perform a GET request (parameter KEY ********** is received from user office on the service website):
http://api.s-shot.ru/PNG/KEY**********/?https://www.google.com.ua

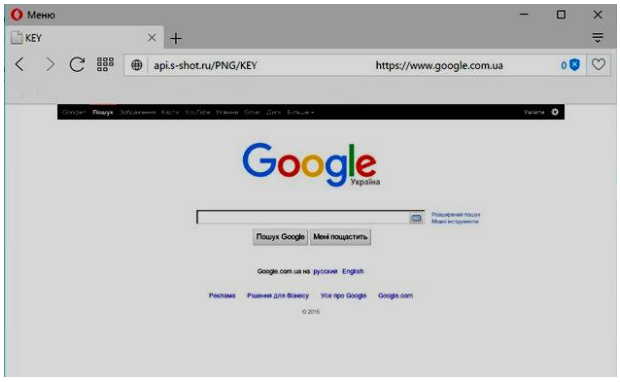

Fig. 1. Images received in response to a request (PNG format)

Another feature is a necessity to set the zoom parameter that is to transfer immediately after transmitting the picture size. This parameter specifies the width of the final image, and thus directly affects the quality of the image [10, 19].

THE THIRD PROBLEM: PHOTO STORING

There are many solutions to this problem, there are some approaches [11–13]:

1. To save screenshots to the server, where the main script file is placed.

An important advantage of this option is no need to create additional requests from third-party services. On the other hand, the drawback consists in necessity to limit and organize convenient access to a gallery, so that requires significant effort and resources.

2. To use a “Dropbox” cloud service profile. This method is instructive concerning an organization and requiring no server resources with a script, although it will later turn out to be that many steps are required to set connection to the server. Furthermore, necessity of using personal profile is not convenient.

3. To use “Google Drive” service. However, this option is difficult to implement, it allows to take all the files under control via requests.

Despite all the difficulties, third way was chosen for the work.

It is inappropriate way to consider such cloud services as “Yandex Disk”, “Cloudinary”, because, according to statistics, “Google” is the most popular and the most commonly used server [16].

In the example given, OAuth2 standard is used and on a recommendation for disk accession two JSON files are used: the first includes information for application authorization, the other consists of access tokens. The first is generated by “Google” service; another is created and used by the script that stores tokens directly. At this stage, an unexpected problem has occurred: there is a lack of connection and periodic repeated authorization difficulties. Therefore, in the final version another way was implemented, which uses the key service profile.

API key It is used for connecting and verification quotas. For example, Google Translate API
Key Account Service It is used by robot account for authorization between servers. Works in Google Cloud API
Creating account data master Several questions about the type of accounting data to be used.

Fig. 2. A key service profile

Unlike the first version, for accession through the key service it is enough to download the generated file with information for connection and specify it in the code.

An example of a code that connects and downloads a test file:

```
$credentialsFile = __DIR__ . '/service.json';
// Initialization of main class to work with Google API
$client = new Google_Client();

// Setting path of the file which contains authorization data
$client->setAuthConfig($credentialsFile);
// Setting application name
$client->setApplicationName("Service Account Example");

//  Setting service that will be involved in interaction
$client->setScopes(Google_Service_Drive::DRIVE);

// Setting a user who will execute actions
$client->setSubject('s***.s***@gmail.com');

// Creating an exemplar of service for working with Google Drive
$service = new Google_Service_Drive($client);

// Creating Google Drive file object
$file = new Google_Service_Drive_DriveFile();

// Creating file description
$file->setDescription('File description');

// Setting the file name under which the file will be saved
$file->setName("File Name.png");

// Setting a unique folder ID where the file is to be saved
$file->setParents([$folderId]);

// Uploading the file
    $createdFile = $service->files->create($file, array(
       'data' => file_get_contents('test_image.png'),
       'mimeType' => 'image/png',
       'uploadType' => 'multipart'
    ));
```

Note that setScopes method takes service address, which you want to access, as an argument, address can be set manually or as a constant class variable. Another, no less important method, called setSubject, where email address is indicated as an argument. While testing it was noticed that there is need to specify the file extension in method setName. In case the file name is given without extension, “Google” service downloads file under the name “Untitled” [17]. Library, provided by service, also has such method as setParents, which indicates the folder where downloaded file is to be saved. In your cloud drive an array of unique folder identifiers serves as function arguments, therefore the file will be placed in each folder specified in the array.

PRACTICAL SOLUTION

Consider the schematic plan of the work.

After getting the request from “Vkontakte” and receiving information, the image URL is formed based on photo_owner_id and photo_id. Later using a screenshot URL the link for request is formed and received by a standard function of PHP file_get_contesnts.

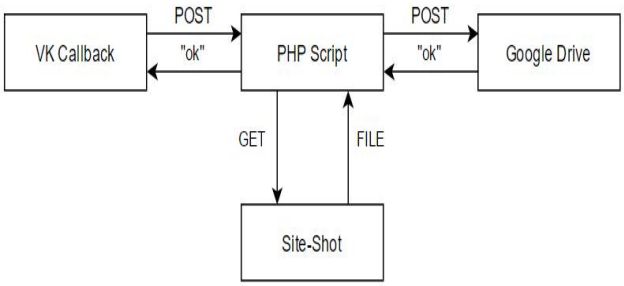

Fig. 3. The scheme receiving requests from applications “Vkontakte”

As a result of these actions, the *$screen* variable contains finished screenshot, which will later be downloaded to “Google Drive” as a simple code.

```
// Getting the data that has been received by request
    $data = json_decode(file_get_contents('php://input'));

    // Receiving main information
    $comment_id = $data->object->id;
    $photo_owner_id = $data->object->photo_owner_id; // ID of the owner of the photo
    $photo_id = $data->object->photo_id; // Photo ID (bet)
    $from_id = $data->object->from_id; // ID of the user that has placed the bet

    // Formation of the image URL
    $photoUrl = 'https://vk.com/photo' . $photo_owner_id . '_' . $photo_id;

    // Forming an URL to send a request to a picture service
    $screenBaseUrl = "http://api.s-shot.ru/KEY********/1300x1000/1500/D1/JS0/png/?";
    $screenUrl = $screenBaseUrl.$photoUrl;

    // Receiving file content (screenshot) via formed URL
    $screen = file_get_contents($screenBaseUrl);

    // Uploading file to cloud storage
    $file = new Google_Service_Drive_DriveFile();
    $file->setDescription('Нова ставка ' . date("d F Y H:i:s") . ' from user id' . $from);
    $file->setMimeType('image/jpeg');
    $file->setName($name . "_" . time() . ".png");
    $file->setParents([$folderId]);
    $createdFile = $service->files->create($file, array(
       'data' => file_get_contents($screenUrl),
       'mimeType' => 'image/png',
       'uploadType' => 'multipart'
            ));
```

When saving (or rather from creating to saving) picture in the “Google Drive” folder, another inaccuracy was found. Time to create the picture was quite large, so “VKontakte” has taken it for a bad request and has duplicate it, because of the every such request has created a picture, it gave an extra load on the server, which was not actually vindicated [16].

To resolve this micro problem, it was used two regular PHP functions popen () and pclose (), as a parameter the first function gets the line that is to be executed in the console and as a result creates a stream object, the second closes the stream object created by the first function, thus overrides expectations of completion of the running script:

```
$fp = popen('nohup php -f sub_script.php &', 'r');
pclose($fp);
```

As a result, one script, which performed receiving request and processing pictures, have turned into two. The first one have worked out feedback of “Vkontakte”, the other have performed creating and uploading pictures on “Google Drive”. After that the working system has been running even better: “VKontakte” gets response about successful processing of the request in time, also the script handles the picture without holding up sending responses to “Vkontakte”.

ANOTHER PROBLEM AND ITS SOLUTION

Created utility will be useful only if the comments placed under the photo are visible without scrolling. In case comments (bets) were many more – only first bets will be visible under the photo, what makes the picture completely useless.

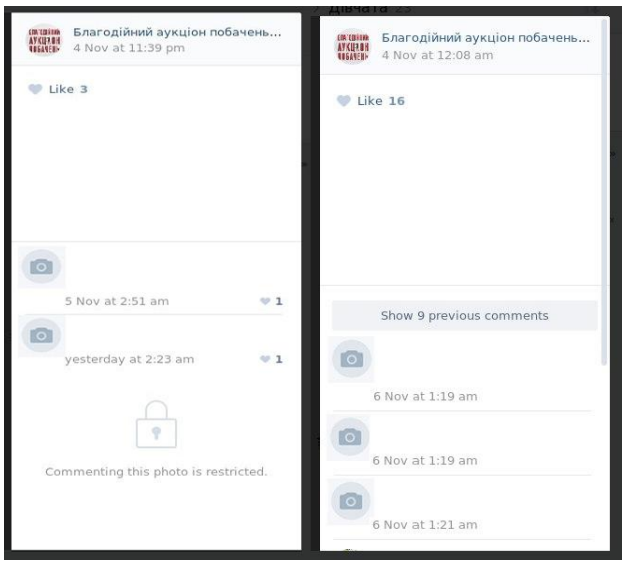

Fig. 4. Useless of the tool in case of a large number of comments

Found on the net it has become a key way in solving the problem, however it was added one more service to the system – URL shortening service. It represents the following: in previous HTML versions (a hypertext markup language) supported such notion as an anchor, it was kind of “#target” link and text of this link points to a unique identifier of the web page element to which you want to scroll. Having analyzed the page with a picture, it was noticed that every comment includes several blocks with identifiers which were generated in a certain way.

Fig. 5. Blocks with identifiers for each comment

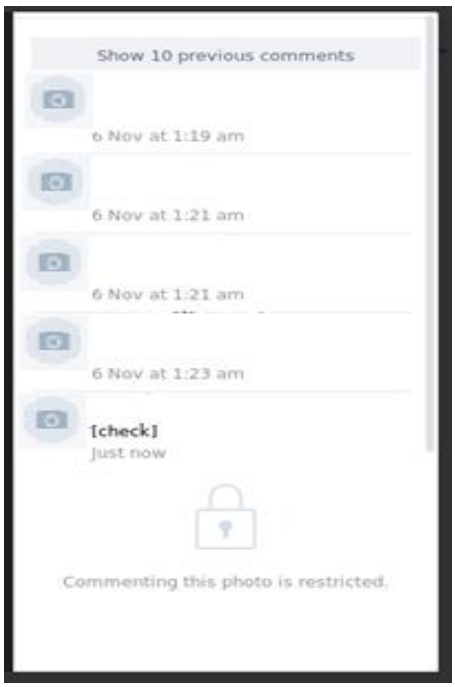

Fig. 6. Query results

Using this identifier in an address line of browser allows to scroll the whole content to the needed comment (bet). However, while generating the link that the screenshot is received by, the anchor sign ("#") is transmitted in the same address line:

http://api.s-shot.ru/PNG/KEY**********/?https://vk.com/photo-46****81_43***** 52 # wpe_bottom-46 *** 781photo_4033

Accordingly. the script is not taken into account by the service, which takes pictures, but by the service, which directly works with it, that is script or browser. Leaving other options aside, as URL shortening server was chosen “Bitly.com”, it has quite loyal limits of the

number of short links. URL was shortened to the following: http://bit.ly/29***Tp

Result:
http://api.s-shot.ru/PNG/KEY**********/?http://bit.ly/29***Tp

With the addition of URL shortening service, the general scheme has evolved:

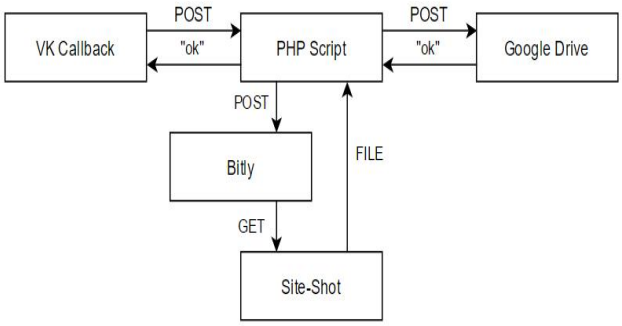

Fig. 7. The general scheme of the system after adding a link shortening service

It is very easy to work with “Bitly”: there are four libraries available, both for developers and service. As the library to work with URL shortening service was chosen one of the simplest:
(https://github.com/Falicon/BitlyPHP). Using this library comes down to two formal actions – “connect and use”:

```
include_once('bitly.php');
$params = array();
$params['access_token'] = '083a815*******7bf84bf';
$params['longUrl'] = 'http://google.com';
$results = bitly_get('shorten', $params);
echo json_encode($results);
```

If successful, the *$ result* array contains everything you need to use:

```
{
  "status_code": 200,
  "status_txt": "OK",
  "data": {
    "long_url": "http:\/\/google.com\/",
    "url": "http:\/\/bit.ly\/2eBHPvW",
    "hash": "2eBHPvW",
    "global_hash": "900913",
    "new_hash": 0
  }
}
```

We are interested only in shortened URL, that is the information in the *$result ['data'] ['url']*. It will be substituted as the link of picture is to be set.

CONCLUSION

As a result, there is a system that saves time, because there is no need to monitor each photo and its comment (bet). All this is done automatically and without your participation. In the process of this system, there is another problem, “VKontakte” sends notifications only when a new comment has appeared, but also it allows authors to edit their comments, because it is sometimes the case that the bet was changed, but the modification was not immediately fixed.

Changing comments may be recorded only in two time points: while creating a comment or adding the next new, since, the changed one gets into the image zone anyway**.** Given the magnitude of the social network, the option of the extension, while changing comments, will be available later, so that will fix not only the bet occurrence but also any action on it.

It is necessary to mention about the separation of the script into two parts while calling the script, which directly performs the picture request and uploads it to the cloud drive, a URL on photos and user ID (for easy search of user who has placed the bet) is passed as cli argument. This information is stored in the field of file description to “Google Drive”.

REFERENCES

1. **Boyko N. 2016** Basic concepts of dynamic recurrent neural networks development / N. Boyko, P. Pobereyko // ECONTECHMOD : an international quarterly journal on economics of technology and modelling processes, Lublin: Polish Academy of Sciences, Vol. 5, No. 2, pp. 63–68.
2. **Leskovec J. 2014** Mining of massive datasets / J. Leskovec, A. Rajaraman, J.D. Ullman, Massachusetts: Cambridge University Press, 470 p.
3. **Mayer-Schoenberger V. 2013** A revolution that will transform how we live, work, and think / V. Mayer-Schoenberger, K. Cukier, Boston New York, 230 p.
4. **Boyko N. 2016** A look trough methods of intellectual data analysis and their applying in informational systems / N. Boyko // Computer sciences and information technologies CSIT 2016 : Proc of XI International scientific practical conference CSIT 2016 : proceedings, Lviv: Lviv Polytechnic Publishing House, pp. 183–185 (in Ukrainian)
5. **Maass W. 2002** Real-time computing without stable states: a new framework for neural computations based on perturbations / W. Maass, T. Natschger, H. Markram / Neural Computation : proceedings, Switzerland: Institute for Theoretical Computer Science, Vol. 11, pp. 2531–2560.
6. **Schrauwen B., Verstraeten D., Campenhout J. V. 2007** An overview of reservoir computing theory, applications and implementations / B. Schrauwen, D. Verstraeten, J. V. Campenhout // Proc. of the 15th European Symp. on Artificial Neural Networks: proceedings, Belgium: Bruges, pp. 471–482.
7. **Coombes S. 2005** Waves, bumps, and patterns in neural field theories / S. Coombes // Biological Cybernetics : proceedings, Nottingham: University of Nottingham, Vol. 93, No. 2, pp. 91–108.
8. **Antonopoulos N. 2010** Cloud Computing: Principles, Systems and Applications / Nick Antonopoulos, Lee Gillam, L.: Springer, 379 p.

9. **Shyshkin V. M. 2011** Safety of cloud computing – problems of risk analysis / V. M. Shyshkin// International scientific practical conference “Automated systems of management and modern information technologiesi”, Tbilisi: Publication House “Technical University”, pp. 142 (in Ukrainian)
10. **Nandkishor G. 2012** Use of cloud computing in library and information science field / Nandkishor Gosavi, Seetal S. Shinde, Bhagyashree Dhakulkar // International Journal of Digital Library Services,Vol. 2, Iss. 3, pp. 51–60, Mode of access: http://www.ijodls.in/uploads/3/6/0/3/3603729/vol._2_july_-_sept._2012_part-2.pdf.
11. **Sangeeta N. 2013** Dhamdhere. Cloud Computing and Virtualization / Sangeeta N. Dhamdhere, 385 p.
12. **Monirul Islam M. 2013** Necessity of cloud computing for digital libraries: Bangladesh perspective / M. Monirul Islam // International Conference on Digital Libraries (ICDL): Vision 2020: Looking Back 10 Years and Forging New Frontiers, pp. 513–524.
13. **Avetysov M. A.** Oblachnыe vыchyslenyya dlya byblyotek / M. A. Avetysov, Mode of access: http://www.aselibrary. ru/blogs/archives/997/.
14. **Mell P. 2011** The NIST Definition of Cloud Computing : Recommendations of the National Institute of Standards and Technology / Peter Mell, Timothy Grance, Mode of access: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.
15. Microsoft support cloud computing, Mode of access: http://www.dw.com/uk/microsoft-531253.
16. **Hewitt C. 2008** ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing / Carl Hewitt // IEEE Internet Computing, Vol. 12, No. 5, pp. 96–99.
17. **Foster I. 2001** The Anatomy of the Grid: Enabling Scalable Virtual Organizations / I. Foster // International Journal of High Performance Computing Applications, Vol. 15, No. 3, pp. 200–222.
18. **Matov O. Ia. 2004** Information technology and the development of GRID systems in high-performance, globally-distributed computing infrastructures corporate cooperation / O. Ia. Matov // Registration, storage and processing of data, V. 6, No. 1, P. 85–98.
19. **Graham S. 2005** Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, SAMS / S. Graham, 816 p.
20. **Foster I**. **2008** Cloud computing and grid computing 360-degree compared / I. Foster, Y. Zhao, I. Raicu, S. Lu // Grid Computing Environments Workshop, GCE'08, pp. 1–10.