Application of Cloud Services for Processing of Information Flows

Nataliya Boyko and Yurii Kryvenchuk

November 20, 2019

Application Of Cloud Services For Processing Of Information Flows

Nataliya Boyko, Yurii Kryvenchuk
Lviv Polytechnic National University
Lviv, Ukraine
nataliya.i.boyko@lpnu.ua
yurii.p.kryvenchuk@lpnu.ua

Abstract—The paper describes the term of cloud storages. It also describes one of the practical approaches to storing the needed information in cloud storages. Theoretical research has been held, advantages and disadvantages of different approaches has been described. The most efficient way of solving the problem was implemented. Problems that might come up in a process of working with social networks were highlighted. The way of problems solving were outlined.

Keywords— cloud storage, screenshot, cloud service, API (Application Programming Interface), software.

I. INTRODUCTION

Cloud storages became popular and widely used starting from year 2006 for a number of reasons, which a key factors and describe this model of storing data:

Physical location of storage is not limited to a single server; data can be accessed from any location in the world, the read/write speed can be better, if the infrastructure has been designed correctly. A cloud storages usage has a number of advantages when comparing to the conventional way of storing data. Users have the ability to purchase service of a certain provider and choose the needed configuration. The important aspect is that cloud storages are highly customizable, so in case of a need, the configuration can be easily changed to suit the needs of a client. Thereby, usage of cloud storages allows clients to save money on their data storage.

There are a big number of different cloud storages providers, however the ways to access them are differ. Generally accepted is the approach of using API (Application Programming Interface), which also might be different in every provider, though, the documentation is almost always available to general public.

It’s worth mentioning that the decentralization of storing data might also have a negative aspect to it. Firstly, users cannot be completely sure that their data is being kept private when they give it to 3rd-party service providers. Secondly, the overall reliability of the service fully depends of the service provider, which in theory could cause some problems

Consequently, cloud storages are a good alternative to traditional ways of storing data, for a number of reasons, such as reliability, scalability, and ease in configuration.

This article describes an approach of creating screenshots of certain publication and comments in a social network “Instagram” and saving them in cloud storage. This process could be beneficial in a case of a need of monitoring some publication, for example, an online auction.

II. SETTING THE TASK

Formulation of the problem: automate process of creating screenshots of a publication and its comments.

According to the task, there are 4 key problems:

1. Determine the quantity of comments and save this value to a database.

2. Perform a check if a new comment has been created.

3. If so, make a screenshots of the publication and comments.

4. Save the screenshot to cloud storages (where and how?)

III. PROBLEM #1 AND #2: CHECK IF NEW COMMENT WAS CREATED, DETERMINE THE QUANTITY OF COMMENTS

Social network “Instagram” does not have any callback features, so the check has to be performed manually. However, it allows the access to a number of attributes of a certain publication (ID of publication, author ID, quantity of comment, etc.). Thereby, we can create a local database in which the quantity of comments for certain publication will be stored, and later we can compare the value from the database to an actual number of comments. If those values differ from each other, we will take a screenshot of the publication and its comments. Acquiring the quantity of comments was performed using a Python module called Instagram API (https://github.com/LevPasha/Instagram-API-python).

Example of a query for receiving the quantity of comment of a last publication:

```
From Instagram API import Instagram API


ifAPI.getUserFeed(AUCTION_PROFILE_ID):
   API = InstagramAPI(settings.LOGIN, settings.PASS)
```

```
API.login()
item = API.LastJson["items"][0]
comment_count = item["comment_count"].
```

IV. Problem #3: Making a screenshot

Several approaches to solve this problem have to be outlined:

1. Using a local server.

2. Using some 3rd-party services.

Selenium Web Driver was chosen as a solution to the problem. Selenium Web Driver is software, which allows controlling the behavior of a web browser. This module has a big variety of different features, however we are interested in ability to make a request, execute some JavaScript code, and making a screenshot.

Example of code to make a screenshot of google.com.ua:

```
From selenium import web driver

driver = webdriver.Chrome(CHROMIUM_DRIVER_PATH)
driver.get(“https://www.google.com.ua/”)
driver.save_screenshot(“screenshot1.png”)
```

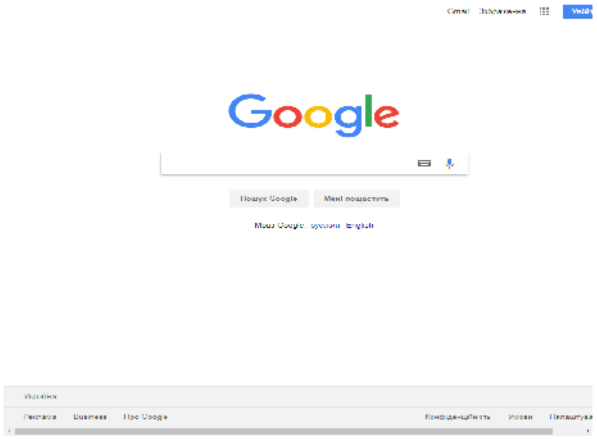


Fig. 1. Screenshot of google.com.ua (*.png extension)*

Execution of JavaScript code is implemented using a function driver.execute_script(), the argument of which is a string with JavaScript code. This is used to hide useless elements of web page.

V. Problem #3: Storing screenshots

There are several issues of solving this problem; some of them are outlined below:

1. Storing screenshots on a local server, with the main script file. The big advantage of such approach is that there is no need to make additional requests to 3rd-party services. Though, the disadvantage is that we need to limit the access to the server while providing easy to use access to the screenshots, which requires a lot of efforts and resources.

2. Using cloud storage provider “Dropbox”. This option is easier from a perspective of configuration and usability. However, as was discovered later, the configuration process is still quite complicated and requires a number of iterations. In additional, usage of personal account is not fully efficient.

3. Using cloud storage service “Google Drive”. This option allows full control of files and folders using requests, however is quite tiresome in implementation.

Despite that, 3rd option was chosen because of its benefits in usage.

The documentation, examples of implementation and other materials on how to use Google Drive is provided on this website - https://developers.google.com/drive/v3/web/quickstart/python

After the activation of Google Drive API, client_secret.json will be created, which has to be moved in a project directory. In order to simplify the process of working with Google Drive API, Python module called pydrive (https://github.com/gsuitedevs/PyDrive) was chosen. It provides a simple interface for making requests, thereby, we don’t have to create requests manually. Documentation of pydrive is provided on this website - https://pythonhosted.org/PyDrive/

The process of authentication is also simplified by pydrive module and is performed automatically in the presence of client_secret.json in working directory.

Example of clients_secret.json:

```
{
    "web": {
        "client_id": "123456789011-ma5shc650k2ns0dmnu6876rud8orfonq.apps.googleusercontent.com",
        "project_id": "cloudservices-199507",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://accounts.google.com/o/oauth2/token",
        "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
        "client_secret": "5ATC9NETaYjgb7PwJkG4rqSU",
        "redirect_uris": ["http://localhost:8080/"],
        "javascript_origins": ["http://localhost:8080"]
    }
}
```

Example of code which establishes connection and uploads a test file:

```
frompydrive.authimportGoogleAuth
frompydrive.driveimportGoogleDrive
```

```
# creation of object of authentication
gauth = GoogleAuth()

# creation of object required to work withGoogleDrive API
drive = GoogleDrive(gauth)

# uploading the file
file_entity = drive.CreateFile()
file_entity.SetContentFile(file_path)
file_entity[‘title’] = file_name
file_entity.Upload()
```

VI. Practical Solution of the Problem

Schematic plan of the solution:

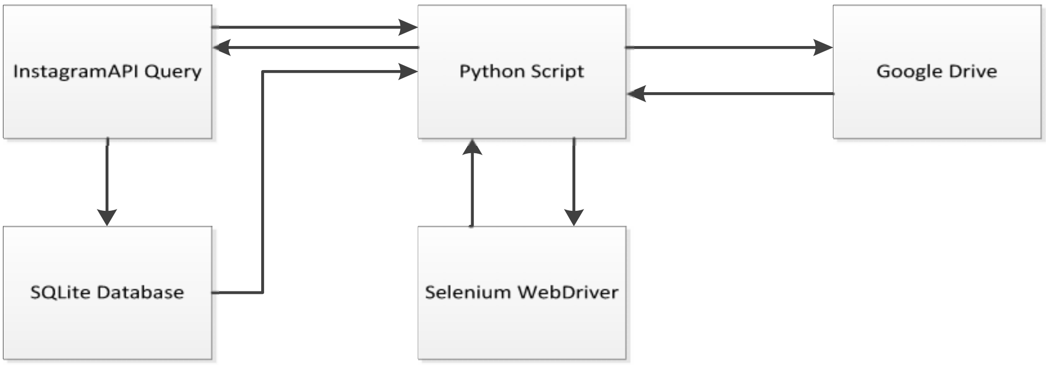


Fig. 2. Schematic plan

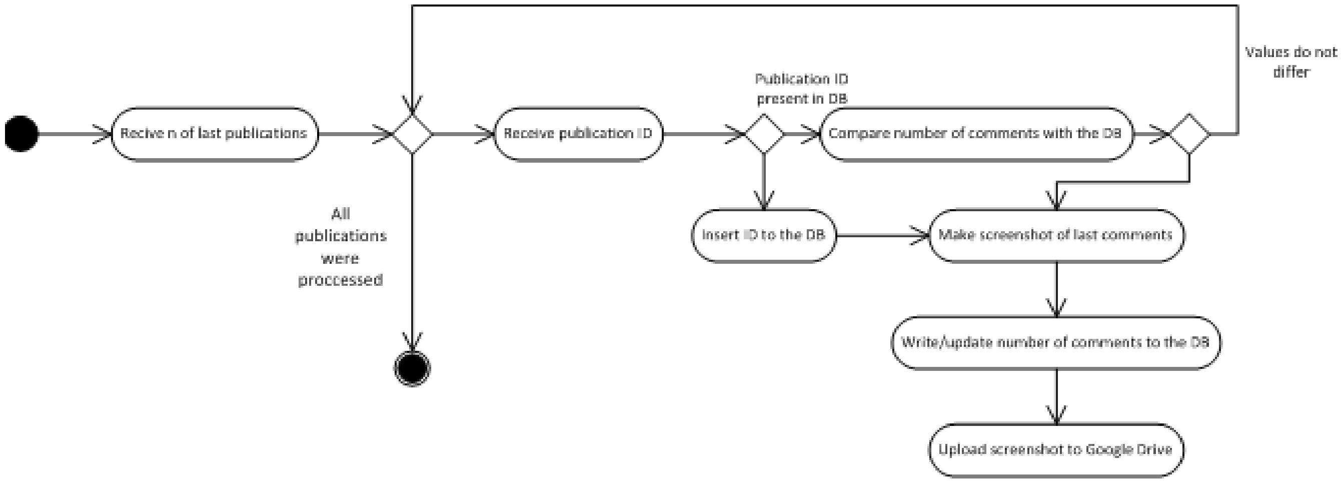
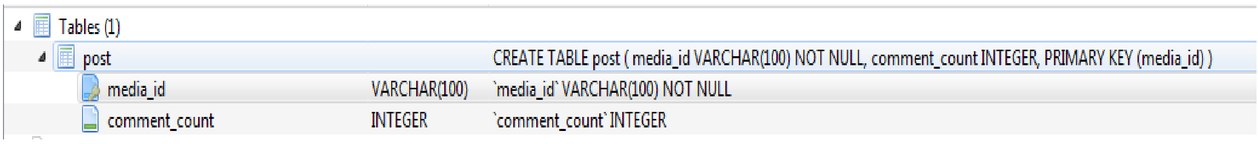

Values do not differ
Publication ID present in DB
Recive n of last publications
Receive publication ID
Compare number of comments with the DB
All publications were proccessed
Insert ID to the DB
Make screenshot of last comments
Write/update number of comments to the DB
Upload screenshot to Google Drive

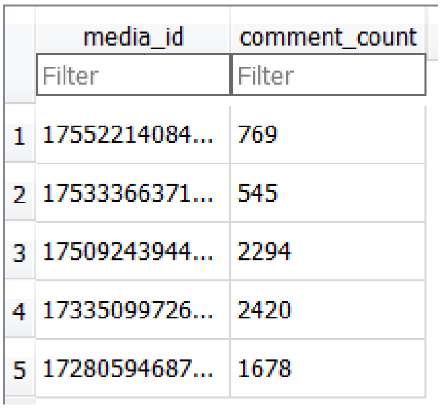
Fig. 3. UML diagram

First of all, we perform a comparison, in order to know if we need to make a screenshot. In order to do that, we receive the number of comments of certain publication and compare that value with the value in database. If those values are different, we call a function, which makes a screenshot. The value in the database is updated soon.



Tables (1)		
post		CREATE TABLE post (media_id VARCHAR(100) NOT NULL, comment_count INTEGER, PRIMARY KEY (media_id))
media_id	VARCHAR(100)	`media_id` VARCHAR(100) NOT NULL
comment_count	INTEGER	`comment_count` INTEGER

Fig. 4. Structure of database



	media_id	comment_count
	Filter	Filter
1	17552214084...	769
2	17533366371...	545
3	17509243944...	2294
4	17335099726...	2420
5	17280594687...	1678

Fig. 5. Example content of database

Function which makes a screenshot and uploads it to Google Driver:

```
defsave_page_screenshot(media_ids):


now = datetime.datetime.now()


withChromeDriverWrapper() asdriver:
withtempfile.TemporaryDirectory() astemp_dir:
formedia_idinmedia_ids:
url = settings.INSTAGRAM_URL % media_id_to_code(media_id)
driver.get(url)
driver.execute_script(JS_SCRIPT)


file_name = "{}.{}".format(now.strftime("%d %B %Y %H-%M-%S"), settings.IMAGE_EXTENSION)
file_path = os.path.join(temp_dir, file_name)
  logger.info("Savingscreenshot %s totempfolder", file_name)
driver.save_screenshot(file_path)


file_entity = drive.CreateFile()
file_entity.SetContentFile(file_path)
file_entity['title'] = file_name
file_entity.Upload()
```

The name of the file is presented in following format for the ease of reading by a user -"%d %B %Y %H-%M-%S".

Another important aspect is usage of tempfile.TemporaryDirectory(). This function creates a temporary directory in which screenshots are saved for the time of execution of the script. After execution has been completed, the directory and its content will be deleted/

JavaScript code used for hiding useless elements of website:

```
varpopup = document.getElementsByClassName('_2pnef');


try {
popup[0].style.visibility = 'hidden';
} catch (e) {
  console.log(e);
}


varcomments = document.getElementsByTagName('ul')[0];
var i = comments.childNodes.length;
while (i--) {
comments.appendChild(comments.childNodes[i]);
}
```

VII. Multiprocessing

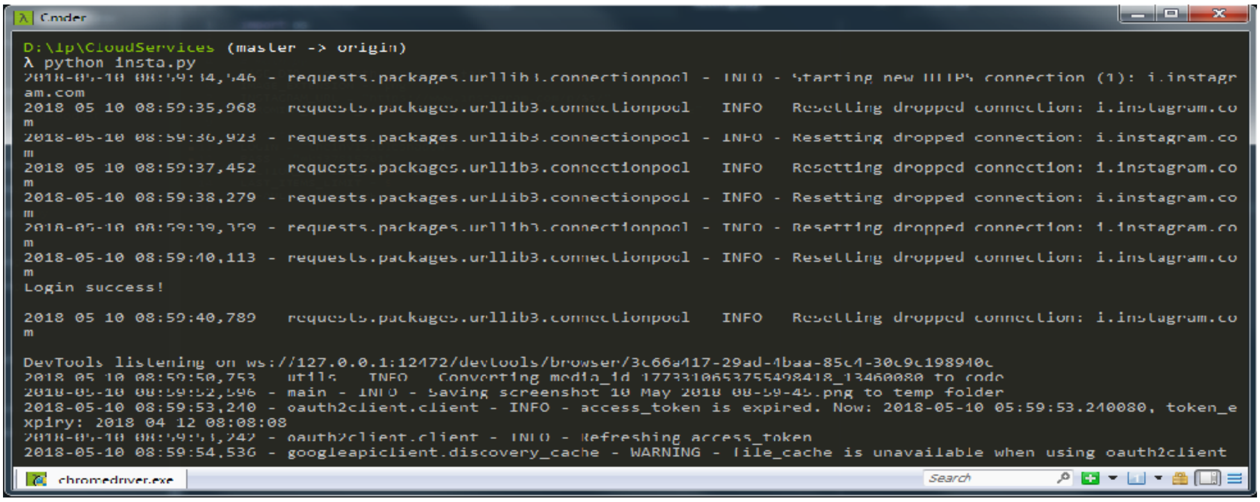
Multiprocessing was used in order to perform asynchronous creation of screenshots. It allows creation of additional processes in order to optimize the execution and the resources need, and also make script work faster. Technically, multiprocessing is a module in Python which allow parallelization of tasks and their execution.

Example of usage:

```
from multiprocessing import Pool


pool = Pool(initializer=init_worker)
pool.apply_async(save_page_screenshot, args=(ids,)).get(timeout=999999)
```

VIII. Results



```
D:\lp\CloudServices (master -> origin)
λ python insta.py
2018-05-10 08:59:34,546 - requests.packages.urllib3.connectionpool - INFO - Starting new HTTPS connection (1): i.instagr
am.com
2018-05-10 08:59:35,968 - requests.packages.urllib3.connectionpool - INFO - Resetting dropped connection: i.instagram.co
m
2018-05-10 08:59:36,923 - requests.packages.urllib3.connectionpool - INFO - Resetting dropped connection: i.instagram.co
m
2018-05-10 08:59:37,452 - requests.packages.urllib3.connectionpool - INFO - Resetting dropped connection: i.instagram.co
m
2018-05-10 08:59:38,279 - requests.packages.urllib3.connectionpool - INFO - Resetting dropped connection: i.instagram.co
m
2018-05-10 08:59:39,359 - requests.packages.urllib3.connectionpool - INFO - Resetting dropped connection: i.instagram.co
m
2018-05-10 08:59:40,113 - requests.packages.urllib3.connectionpool - INFO - Resetting dropped connection: i.instagram.co
m
Login success!

2018-05-10 08:59:40,789 - requests.packages.urllib3.connectionpool - INFO - Resetting dropped connection: i.instagram.co
m

DevTools listening on ws://127.0.0.1:12472/devtools/browser/3c66a417-29ad-4baa-85c4-30c9c198940c
2018-05-10 08:59:50,753 - utils - INFO - Converting media_id 1773310653755498418_13460080 to code
2018-05-10 08:59:52,596 - main - INFO - Saving screenshot 10 May 2018 08-59-45.png to temp folder
2018-05-10 08:59:53,240 - oauth2client.client - INFO - access_token is expired. Now: 2018-05-10 05:59:53.240080, token_e
xpiry: 2018 04 12 08:08:08
2018-05-10 08:59:53,242 - oauth2client.client - INFO - Refreshing access_token
2018-05-10 08:59:54,536 - googleapiclient.discovery_cache - WARNING - file_cache is unavailable when using oauth2client
```

Fig. 6. Execution of the script

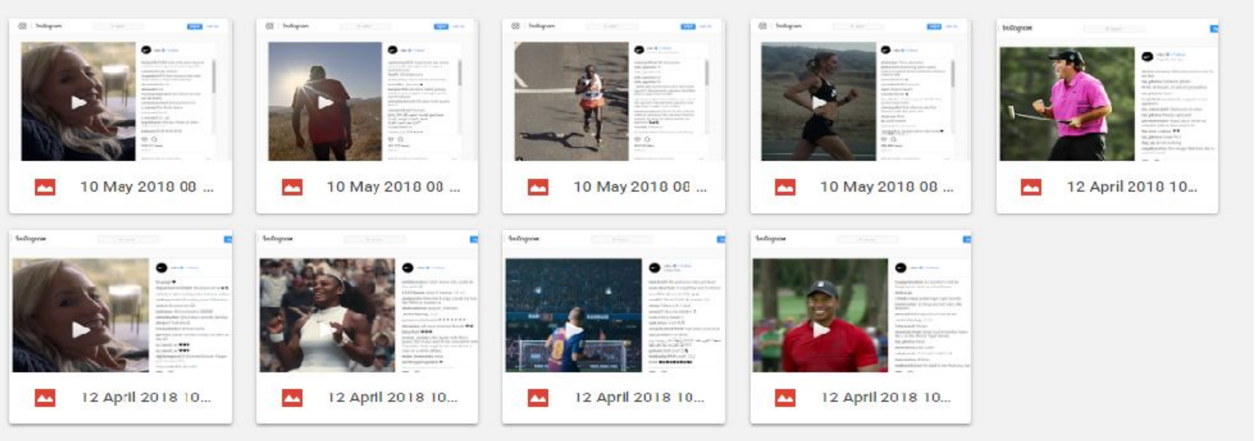


Fig. 7. Content of the Google Drive folder after the execution

IX. Conclusions and Perspectives of Further Scientific Developments

As a result of this experiment, we were able to achieve a system, which saves time, since there is no need to manually track every publication and its comments. This process is performed automatically and without interference of a human. Because of multiprocessing, the script works in an optimal and fast way and doesn't require large resources. Current system is reliable, easy to use and configure. Also, this system is easy to modify and/or make some adjustments to it, since all key configuration variables are stored in settings.py file, such as extension of the screenshot, Instagram login and password of a user, timeout between the checks, etc.

References

[1] N. Sangeeta. *Cloud Computing and Virtualization*, Dhamdhere, 2013, 385 p.

[2] M. Monirul Islam. “Necessity of cloud computing for digital libraries: Bangladesh perspective”, International Conference on Digital Libraries (ICDL) 2013 : Vision 2020 : Looking Back 10 Years and Forging New Frontiers, pp. 513–524.

[3] V. Estivill-Castro and I. Lee. “Amoeba: Hierarchical clustering based on spatial proximity using Delaunay diagram” [9th Intern. Symp. on spatial data handling, Beijing, China, 2000, pp. 26–41].

[4] H.-Y. Kang and B.-J. Lim and K.-J. Li. *P2P Spatial query processing by Delaunay triangulation*. Lecture notes in computer science, vol. 3428,Springer/Heidelberg, 2005, pp. 136–150.

[5] C. Boehm and K. Kailing and H. Kriegel and P. Kroeger. “Density connected clus-tering with local subspace preferences” IEEE Computer Society [Proc. of the 4th IEEE Intern. conf. on data mining, Los Alamitos, 2004, pp. 27–34].

[6] D. Harel and Y. Koren. “Clustering spatial data using random walks”. Proc. of the 7th ACM SIGKDD Intern. conf. on knowledge discovery and data mining, San Francisco, California, 200, pp. 281–286.

[7] A.K Tung and J. Hou and J. Han. “Spatial clustering in the presence of obstacles”. The 17th Intern. conf. on data engineering (ICDE’01), Heidelberg, 2001, pp. 359–367.

[8] O.I. Veres and N.B. Shakhovska. “Elements of the formal model big date”. The 11th Intern. conf. Perspective Technologies and Methods in MEMS Design (MEMSTEH), Polyana, 2015, pp. 81-83

[9] R. Agrawal and J. Gehrke and D. Gunopulos and P. Raghavan. *Automatic sub-space clustering of high dimensional data*. vol. 11(1), Data mining knowledge discovery, 2005, pp. 5–33.

[10] L. Guimei and L. Jinyan and K. Sim and W. Limsoon. “Distance based subspace clustering with flexible dimension partitioning” [Proc. of the IEEE 23rd Intern. conf. on digital object identifier, vol. 15. Iss. 20, 2007, pp. 1250–1254].

[11] C. Aggarwal and P. Yu. “Finding generalized projected clusters in high dimensional spaces”. ACM SIGMOD Intern. conf. on management of data, 2000, pp. 70–81.

[12] C.M. Procopiuc and M. Jones and P.K. Agarwal and T.M. Murali. “A Monte Carlo algorithm for fast projective clustering”. ACM SIGMOD Intern. conf. on management of data, Madison, Wisconsin, USA, 2002, pp. 418–427.

[13] M. Ankerst and M. Ester and H.-P. Kriegel. “Towards an effective cooperation of the user and the computer for classification” [Proc. of the 6th ACM SIGKDD Intern. conf. on knowledge discovery and data mining, Boston, Massachusetts, USA, 2000, pp. 179–188].

[14] D.J. Peuquet. *Representations of space and time*. N. Y.: Guilford Press, 2002.

[15] D. Guo and D.J. Peuquet and M. Gahegan. *ICEAGE: Interactive clustering and exploration of large and high-dimensional geodata*, vol. 3, N. 7, Geoinfor-matica, 2003, pp. 229–253.

[16] N.I. Boyko. “A look trough methods of intellectual data analysis and their applying in informational systems”. [In Scientific and Technical Conference “Computer Sciences and Information Technologies (CSIT), IEEE, XIth International, 2016, pp. 183-185].

[17] N.I. Boyko and N.B. Shakhovska and T. Sviridova. “Use of machine learning in the forecast of clinical consequences of cancer diseases” [In 7th Mediterranean Conference on Embedded Computing, IEEE MECO’2018, pp. 531-536].

[18] N.I. Boyko. “Advanced technologies of big data research in distributed information systems”, Radio Electronics, Computer Science, Control. № 4, Zaporizhzhya: Zaporizhzhya National Technical University, 2017, pp. 66-77.

[19] N.B. Shakhovska and O.B. Vovk and R.T. Hasko and Yu.P. Kryvenchuk. “The Method of Big Data Processing for Distance Educational System” [In Conference on Computer Science and Information Technologies, pp. 461-473]

[20] N.B. Shakhovska and O.B. Vovk and Yu.P. Kryvenchuk. *Uncertainty reduction in Big data catalogue for information product quality evaluation*. Eastern-European Journal of Enterprise Technologies, vol 1, No 2 (91), 2018, pp. 12-20.