GREEN MOBILE APPLICATION DEVELOPMENT THROUGH SOFTWARE LOCALIZATION

Nikolay Kasakliev, Elena Somova, Margarita Gocheva

University of Plovdiv "Paisii Hilendarski"
e-mails: kasakliev.pu@gmail.com, eledel@uni-plovdiv.com,
margarita.gochev@gmail.com,
Bulgaria

Abstract: Developing green mobile application software is closely connected to reducing energy consumption during the applications' usage. The article demonstrates an initial easy-to-use approach that can affect energy efficiency through appropriate localization of mobile applications. It presents existing energy-aware techniques, tools, and repositories, as well as the types of mobile applications and the nature of the localization process for each type. It will also guide developers through best practices and recommendations, depending on the type of application and mobile platform.

Keywords: localization, mobile application, green software, energy consumption

1.INTRODUCTION

Nowadays, there is an increased use of mobile devices and mobile applications (MA) in various daily activities – communication, business, entertainment, training and more. As a result of this tendency, developers of MAs are being challenged to create applications that have very little environmental impact, or in other words to create a so-called green software. One of the early stages of a MA development that can lead to energy saving consumption without the need for software code optimization and other more difficult approaches, is localization. Localization aims to respond more quickly to consumer demands for a better user experience, no matter of age or different digital and linguistic culture. Depending on the type of mobile application and platform, localization can be done differently using various tools and approaches – from different language versions, through the use of resource and configuration files, to machine translation.

The purpose of this study is to demonstrate the importance of green software development (including for mobile devices) and to propose a single approach –

through appropriate localization, where, in a few easy steps, large energy efficiency effects can be achieved.

Section 2 presents the state of green software technology research. Section 3 reveals the methodology of the research. Sections 4 and 5 present the mobile applications types and the nature of the localization process. Section 6 uncovers the different approaches that can be used to localize mobile applications. The article concludes with suggested practices and recommendations for developers, depending on the type of application and mobile platform.

2. GREEN SOFTWARE

Recently, there has been an increased attention to the efficiency and effectiveness of software, the so-called green software. [1] gives the definition of green software as "Computer software that can be developed and used efficiently and effectively with minimal or no impact on the environment".

How green is a software is determined by software reduction of energy consumption? In his PhD thesis [2], Procaccianti concludes: “Energy consumption is software-defined. There is no one-size-fits-all. No improvement is possible without measurement.”

Talking about mobile software, energy consumption is a main indicator for the performance of the devices. Many authors work on measurement of energy consumption on mobile devices with different metrics. [3] studies (measuring and analyzing) energy consumption of software running on mobile devices. Their results show that besides the difference in the number of features covered by the software, also their implementation plays an important role in energy consumption. [4] presents how to set up a software engineering lab session that focuses on energy efficiency measurement during software testing at different levels.

In the previous century, the software engineer’s main goal was to develop fast software systems. Nowadays this turned to making low energy consumption devices which is a main goal not only for hardware producers, but also for software developers. Some authors study programming languages and written program code with different approaches in order to show how energy-efficient software should be created.

[5] defines a ranking of energy efficiency in programming languages (twenty-seven well-known software languages) on the base of monitoring the energy consumed during execution of a set of computing problems, implemented in these different programming languages. Their results show that although the fastest languages tend to be the lowest consuming ones, there are some cases where slower languages are more energy efficient than faster ones. [6] also shows how memory usage influences energy consumption.

[7] proposes a technique to detect energy inefficient fragments in the source code of a software system.

[8] studies the energy consumption of queries in data centers in companies, both in relational and non-relational database approaches.

A little tool support exists to understand the energy consumption of a software system and/or automatically to improve its code. [9] presents a tool jStanley, which automatically finds collections in Java programs that can be replaced by others with a positive impact on the energy consumption as well as on the execution time.

[10] presents a technique and a prototype tool to statically estimate the worst case energy consumption for all products in a software product line, without having to produce, run and measure the energy in all of them.

[11] implements the GreenSource infrastructure with a large body of open source code, executable Android applications (incl. from MUSE Java source code repository) and curated dataset, containing energy code metrics. The authors developed the AnaDroid tool which instruments its code, compiles and executes it with test inputs in any Android device, while collecting energy metrics.

Energy efficient approaches that can be taken into account while designing mobile applications are discussed in [12, 13, 14].

First approach is to design energy efficient GUI, for example by using low-energy color schemes, hot keys, quick buttons, user input caches, etc.

Another approach deals with memory and performance and it recommends using techniques like: releasing resources or services as soon as possible, using animations with care, performing multiple operations at ones, etc.

Measuring battery consumption is another approach that is widely used to examine energy efficiency in mobile applications' development. Measuring can be made by software or hardware tools.

Network operations are very important in current mobile applications and developers can achieve energy efficiency by reducing size of transmitted data over network, offline mode functionality or disable heavy data connections over cellular networks.

3. METHODOLOGY

To achieve its purpose, the analytical study is divided into three stages. The first stage gives a brief overview of the types of MA. Native, web and hybrid MAs are being introduced with their features. The second stage examines the different types of MA, defines the localization process and presents some features and advantages of the localization that justifies the need of this process. In the third stage, different approaches, frameworks and localization tools are explored and presented. As a result, the main approaches, features, some good practices and recommendations that can help developers in this process, are outlined. A brief

assessment of the potential positive effect of localization in terms of reducing the energy consumption of localized MAs is made.

4. TYPES OF MOBILE APPLICATIONS

Mobile devices with many different hardware parameters are available on the market. They, in turn, are managed by several mobile platforms (Android, iOS, Windows 10 mobile, Tizen, KaiOS, etc.). Regardless of the platform, MAs developers also have to make decisions about what the applications really are. The choice is driven by a number of factors, including: the need of internet connectivity, working with device sensors, data sharing, the use of third-party information services, etc. There are three available options: native, web, or hybrid applications [15].

Native MA – are installed and operated on the device and are accessible through the user interface. Native apps are most commonly installed through app stores (such as Google Play or the Apple App Store). They are specifically designed for a particular mobile platform, and can benefit from almost all device features - they can use the camera, GPS, accelerometer, compass, contact list, etc.

Mobile Web – are not typical mobile applications, they are websites which look and work like native applications. They are managed by the device browser and typically use HTML5 and JavaScript along with server-side web development technologies.

Hybrid MA – like the native applications, they are installed and run on the device, but are developed using web technologies (HTML5, CSS and JavaScript). They work in the so-called native container using web view for HTML visualization and local JavaScript implementation. Unlike mobile web applications, they can use an intermediate abstraction layer to access the device sensors.

5. LOCALIZATION

Localization (l10n) is a process of adapting the language, cultural and technical characteristics of an application, incl. mobile, to the target market. The process involves modifying user interface (texts, images, date format, numbers, currency, etc.) and documentation according to the specific user groups. Localization can include different spoken variations and dialects of the same language, which are referred to as locales.

Localization is implemented in different ways depending on the type of MA and/or the development tools and technologies. These include the use of translation text files, audio files in different languages, images, directly in hardcoded locale strings, through specific software development kits, tools, libraries (e.g. ngx-translate), and even cloud services such as the Google Play App Translation service.

Due to the specificity of mobile devices, in particular the small screen, some linguistic rules must be observed in the localization.

Most illustrative is the example of texts in languages that are read from right to left. In this case, translating the text content will not be enough, and you will need to move the navigation menu to the right, for example. Another feature was mentioned by [16] - the text in French, German and Spanish occupies about 30% more space than the text in English. Therefore, no fixed width and height values should be specified for text interface elements.

Attention should be paid in other specific situations as well, such as when localization is often required by regulatory requirements. For instance, in the Bulgarian Consumer Protection Act (https://lex.bg/laws/ldoc/2135513678) in Art. 4 there exist requirements for disclosure of composition, quantity, price, etc., which allow consumers to make their informed choice. From this example it becomes clear that every trading application must meet the requirements of this law and accordingly provide the information in the appropriate form. For example, for Bulgaria, prices should be announced in BGN (VAT included), weight in grams / kilogram, length in mm / cm, etc.

This research aims to place an emphasis that has and will have, to our belief, an increasing influence on the development of MAs in the future, namely the development of green MAs. When it comes to mobile technologies and their impact on nature, studies such as [17], estimate that this impact is growing. The main factors that influence this impact include energy consumption and recycling of old mobile devices and/or of their components. This idea is shared by a study of [18], which predicts that, by 2020, smartphones will consume more power than PCs and laptops.

In our opinion, localization can reduce energy consumption and then reduce the negative environmental impact. The usage time of localized MAs, whether web, hybrid or native, will be shorter, because the user will not spend time trying to translate texts or convert currency units or other measures. In this regard, there is a possibility for the user to put into service other applications, Internet browsers and online translation services, which will lead to additional energy consumption. The increased usage time of mobile devices also leads to a decrease in battery life due to the great number of charging and discharging cycles. This, in turn, leads to the need to recycle an increasing amount of batteries. However, we should not forget that the recycling process has an environmental impact as well [19]. Next, if the MA is not adequately localized, it increases the risk of multiple inaccuracies (e.g. navigation) and errors, incl. wrong commands (sorting, unwanted deletion of data), which will also increase the usage time and energy consumption.

In addition to the reasons listed above, software developers themselves resort to localizing their products, as they can achieve many benefits for their business. Some of them are:

- Reaching a wider audience, incl. new users when implementing multiple language versions of MA;
- Better reputation for the developer;
- Better understanding of functionality by users;
- Avoiding mistranslation (incl. the resulting wrong format of numbers, dates, currencies, etc.);
- Reducing development and maintenance costs (for example, by avoiding hard-coded strings and using resource or configuration files instead);
- Obtaining a competitive advantage by reducing the software cost and developing the qualitative applications with better user experience.

The negatives or disadvantages are associated with an increase (in some cases) of development costs due to the need to hire additional staff, such as translators, staff in call centers speaking multiple languages, or legal professionals who are familiar with the local law.

6. MOBILE APPLICATION LOCALIZATION APPROACHES

6.1. Mobile Web

Mobile web applications are essentially web applications designed to be used/ rendered appropriately on mobile devices. In localization, methods which are specific for the development technology are applied or a framework is used. Both approaches are explained below.

For **static web applications**, HTML is used for coding. Localization approaches for these applications are very limited and are reduced to creating separate pages for different regions or using JavaScript. For example, in any HTML page, the 'lang' (Fig. 1) or data- * attributes for a particular element can be used, and then with the appropriate JS/JQuery code the corresponding content can be displayed to the client.

```
<p lang = "bg"> Bulgarian localization </p><p lang = "en"> English localization </p>
```

Fig. 1. Listing of an example of using the 'lang' tag

When applying this approach good practices show that while using mobile devices some difficulties might appear, such are restricted or disabled JavaScript code or cookies for older versions of mobile browsers, so we should always check and enable these features.

Tools such as Lingumania [20], which is a JS localization library based on a JSON object, contains data on supported languages and provides translated segments for developers.

With **dynamic web applications**, different approaches are available, depending on the development framework used.

Nowadays, developers have a choice of multiple frameworks which they can use to create dynamic websites. Worldwide usage, according to [21], is distributed as follows: PHP in about 39% of sites, ASP.NET in about 13%, ASP.NET Ajax, J2EE and Ruby on Rails in about 4% and the others according to these statistics in 1 % - 2%.

For example, if the web application is based on PHP, then one of the following methods can be used for localization:

- use of the JSON language files or associative arrays (Fig. 2);
- use of the internationalization extension 'Intl'. It allows operations such as formatting, transliteration, encoding conversion, time zone work, etc. An example of formatting a numeric value using the setlocale function is added in (Fig. 3);

When using ASP.NET framework, resource (.resx) files are used. These files are essentially XML formatted content - strings, image paths, and more in key-value format (Fig. 4). A separate resource file is created for each supported culture.

```
<?php
$GLOBALS["messages"] = array (
 'en'=> array(
  'Monday' => 'Monday',
  'Tuesday' => 'Tuesday',
 ),
 'de'=> array(
  'Monday' => 'Montag',
  'Tuesday' => 'Dienstag',
)
);
```

Fig. 2. Listing of an example of using associative arrays

```
<?php
 setlocale( LC_ALL, 'fr_FR' );
 printNumber();
function printNumber()
 {
  $number = 199.50;
  print $number;
 }
?>
```

Fig. 3. Listing of using the setlocale () function

```
Resource.en.resx
<data name="prompt"
xml:space="default">
 <value>Enter score:</value>
</data>
Resource.fr.resx
<data name="prompt"
xml:space="default">
 <value>Entrer score:</value>
</data>
```

Fig. 4. Listing of resource ASP.NET XML files

6.2. Hybrid MAs

Multiple frameworks can be used to develop hybrid MAs. Some of the current popular ones are Ionic and Framework7. Both frameworks allow the creation of multi-platform MAs based on shared code. Web technologies such as HTML5, CSS and JavaScript are used for development, but other frameworks can be integrated, such as Cordova, React and Vue for Ionic. Ionic is an open source framework for web and MA development across multiple platforms using a single source code. An approach for three-step localization is demonstrated in [22]:

Step 1. Finding the language, either automatically (setting up the browser or mobile device, Fig. 5) [23] or manually, by user's action, such as selecting a language from a drop-down list.

Step 2. Creating language JSON files with translated text (Fig. 6).

Step 3. Implementing the ngx-translate library (Fig. 7).

```
navigator.globalization.getPreferredLanguage(
 function (language) {alert('language: ' + language.value + '\n');},
 function ()  {alert('Error getting language\n');}
);
```

Fig. 5. Listing of finding the device's default language using the Cordova Globalization plugin

```
{
"Header_1": "Hola señor",
"Header_2": "Hola {{value}}",
"data": {"name": "Me llamo {{name_value}}"}
}
```

Fig. 6. Listing of es.json file

```
_initialiseTranslation(): void {
  this._translate.get('Header_1').subscribe((res: string) => {
   this.title = res;
  });
```

Fig. 7. Listing of extraction of TypeScript code for translation

6.3. Native MA

Localization of a native MA is characterized by a specific approach depending on the mobile platform. According to [24], the most common one currently is Android with about 76% market share, followed by iOS with 22%, KaiOS with just under 1% and others even less.

As a dominant mobile platform, Android OS devices operate in many regions, and the great challenge for MA developers of this platform is to provide localization for each culture. To facilitate this process, the platform is designed to allow easy localization. The basic approach is to use resource files. Resources are text strings, audio, layouts, images, and other static data. Each MA can include multiple resources, each adjusted to different configurations (screen size, device orientation, default language, etc.) of the devices. When launching an MA, the platform automatically loads resources that match the configuration of the particular device.

The standard approach involves creating localized resources allocated to separate subfolders of the res / folder. For example, localization of English and Bulgarian texts should create alternative strings.xml files in folders respectively res/values /strings.xml and res/values-bg/strings.xml (Fig. 8). The access to resource (interface string) app_name is possible with the following Java code:

```
String title=getResources().getString(R.string.app_name);
```

```
<resources>
  <string name="app_name">My Application</string>
  </resources>
```

```
<resources>
  <string name="app_name">Моето приложение</string>
</resources>
```

Fig. 8. Listing of sample files res/values /strings.xml and res/values-bg / strings.xml

```
String.toLocaleString({
"en": {
   "%title": "English - l10n.js demo", "%info": "You are viewing ..."
   },
"nl": {
   "%title": "Nederlands - Demo van l10n.js", "%info": "U ziet nu een ..."
   },
});
```

Fig. 9. Listing of toLocaleString() function

An alternative approach may be the use of application translation services. An example of a service of this kind is the App Translation service [25]. The service offers translation of the application resources made by a person for a fee.

Files are also used to locate iOS applications. In this case, they are called Localizable.string and are located in a separate folder for each language. The file structure is key-value, for example: "Delete" = "Effacer";

The third most widespread mobile platform currently is KaiOS. MAs are web-based and a modified version of the l10n.js library is used for localization. It is based on localization files in JSON format (Fig. 9) and the invocation of the toLocaleString() function for each localization string [26].

7. RECOMMENDATIONS

The following best practices and recommendations for developers of MA can be given:

1. When designing an MA, one of the goals to be achieved is the energy efficiency of the application.

2. The choice of a localization approach should be carried out before the actual steps of realization of this activity. The main questions to be answered are how many and which languages should be supported by the MA.

3. The localization planning should be done before the MA design stage, by going through the internationalization process (i18n), so that many natural languages can be maintained. [27] defines the process as "design or modification of software so that each user can choose a language". The result of this process is that the MA is flexible enough to support virtually any language. The recommendation is to make the early stages (requirements analysis, feasibility checks and design) of software development, since localization is then much easier to implement.

4. Extensively test the MA. Due to the wide variety of mobile devices and mobile platform versions, this stage can be very time-consuming and expensive as many types of tests are conducted [28]. From a localization point of view, testing should be done by people with a fluent language, covering the user interface, untranslated strings, error messages, help, etc.

5. The target market for MA must be examined in order to select the appropriate ones (for the region and/or language): the name of the MA, description, subtitle, keywords, etc. This can increase the visibility and the number of downloads and installs of the MA. Millions of MAs are currently available in app stores, often hundreds of one-of-a-kind. In order to overcome competition, software developers need to apply techniques known in SEO – the so-called App Store Optimization (ASO), which according to [29] is "app store marketing or mobile app search engine optimization".

Lack of a good localization, accompanied by poor performance in mobile app stores, may lead to a lot of negative user reviews, which is likely to result in a negative financial impact.

8. CONCLUSION

Nowadays, one common question connected to the influence of energy consumption to the environment is: how to reduce energy consumption when using

mobile applications? Developers start to speak about green software application, that are environmentally friendly.

The article presents existing energy-aware techniques, approaches, and tools.

The main contribution of the article is an easy approach to getting started with the green software creation by using various methods of localization of the MA. Different types of MA are observed and appropriate ways for localization are proposed. Best practices and recommendations, depending on the application type and mobile platform are given.

The work has been inspired by project No 2017-1-SK01-KA203-035402 „Focusing Education on Composability, Comprehensibility and Correctness of Working Software" in the frame of program Erasmus +, Key action 2: Strategic partnerships.

REFERENCES

[1] Murugesan, S., G. Gangadharan. *Harnessing Green IT: Principles and Practices*, ISBN: 978-1-119-97005-7, Wiley-IEEE Computer Society Pr, 2012.

[2] Procaccianti, G. Energy-Efficient Software. *SIKS (Dutch Research School for Information and Knowledge Systems) Dissertation Series* No. 2015-134, Vrije Universiteit Amsterdam (Free University of Amsterdam), 2015.

[3] Szabó, C., E. Alzeyani. Measuring energy efficiency of selected working software. *Studia Universitatis Babes-Bolyai, Informatica* Vol. 63, pp. 5-16. Also presented at 12th Joint Conference on Mathematics and Computer Science, Cluj-Napoca, Romania. June 14-17, 2018.

[4] Szabó, C. Focusing education on energy efficiency measurements during software testing. *Technical report of the Amsterdam Teacher's Training of 3COWS project*, 2018. Available at: https://www.inf.elte.hu/en/dstore/document/1252/Csaba_Szabo.pdf

[5] Couto, M., R. Pereira, F. Ribeiro, R. Rua, J. Saraiva. Towards a green ranking for programming languages. *Proc. of the 21st Brazilian Symposium on Programming Languages*, Fortaleza, Brazil, 2017. Available at: https://greenlab.di.uminho.pt/wp-content/uploads/2017/09/paperSBLP.pdf

[6] Pereira, R., M. Couto, F. Ribeiro, R. Rua, J. Cunha, JP. Fernandes, J. Saraiva. Energy efficiency across programming languages: How do energy, time, and memory relate? *Proc. of the 10th ACM SIGPLAN International Conference on Software Language Engineering, co-located with SPLASH*, Vancouver, Canada, 2017, pp. 256-567. Available at: https://greenlab.di.uminho.pt/wp-content/uploads/2017/10/ sleFinal.pdf

[7] Pereira, R., T. Carção, M. Couto, J. Cunha, JP. Fernandes. Helping programmers improve the energy efficiency of source code. *Proc. of the 39th International Conference on Software Engineering Companion*, Argentina, 2017, pp 238–240. Available at: https://docentes.fct.unl.pt/jmc-cunha/files/paper_7.pdf

[8] Saraiva, J., M. Guimarães, O. Belo. An economic energy approach for queries on data centers. *Proc. of 3rd International Conference on Energy & Environment: bringing together Economics and Engineering (ICEE'2017)*, Porto, Portugal, June 29-30, 2017. Available at: https://greenlab.di.uminho.pt/wp-content/uploads/2017/06/2017-ICEE-SaraivaEtAl-CRP.pdf

[9] Pereira, R., P. Simão, J. Cunha, J. Saraiva. Jstanley: Placing a green thumb on java collections. *Proc. of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, Montpellier, France, 2018, pp. 856-859.

[10] Couto, M., P. Borba, J. Cunha, JP. Fernandes, R. Pereira. Products go green: Worst-case energy consumption in software product lines. *Proc. of the 21st International Systems and Software Product Line Conf.*, Seville, Spain, 2017, pp. 84-93. Available at: https://docentes.fct.unl.pt/jmc-cunha/files/paper_9.pdf

[11] Rua, R., M. Couto, J. Saraiva. GreenSource: a large-scale collection of Android code, tests and energy metrics. *Proc. of the 16th International Conference on Mining Software Repositories*, Montreal, Canada, 2019, pp. 176-180.

[12] Cruz, L., R. Abreu. Catalog of energy patterns for mobile applications. *J. of Empirical Software Engineering*, **No. 4** (vol. 24), August 2019, pp 2209–2235, Available at: https://doi.org/10.1007/s10664-019-09682-0

[13] Nidawi, H., T. Koh, K. Dawood, A. Khaleel. Energy consumption patterns of mobile applications in android platform: A systematic literature review. *Journal of Theoretical and Applied Information Technology*, 95 (24). 6776 – 6787, December 2017, ISSN 1992-8645; ESSN: 1817-3195. Available at: http://www.jatit.org/volumes/Vol95No9/1Vol95No9.pdf, accessed 20.09.2019.

[14] Vallerio, K., L. Zhong, N. Jha. Energy-efficient graphical user interface design. *J. of IEEE Transactions on Mobile Computing*, **No. 7** (vol. 5), pp. 846-859, July 2006. doi: 10.1109/TMC.2006.97

[15] Kasakliev, N., Current Trends in Mobile Application Development. *J. on Computer Science and Communication*, **No. 2** (vol.4), pp. 96-105, 2015, ISSN: 1314-7846. (in Bulgarian)

[16] Thoms, N. App localization done right. https://appdevelopermagazine.com/app-localization-done-right/, accessed 20.09.2019.

[17] Ylä-Mella, J., E. Pongrácz, P. Tanskanen, R. Keiski. Environmental Impact of Mobile Phones: Material Content. *Proc. of International Conference on Solid Waste Technology and Management*, Philadelphia, PA, USA, 2007.

[18] Belkhir, L., A. Elmeligi. Assessing ICT global emissions footprint: Trends to 2040 & recommendations. *J. of Cleaner Production*, (vol.177), 2018, pp. 448-463.

[19] Boyden, A., Vi Kie Soo, M. Doolan. The Environmental Impacts of Recycling Portable Lithium-Ion Batteries. *Procedia CIRP*, (Vol.48), 2016, pp. 188-193.

[20] *Simple Javascript Based Localization*, http://www.lingumania.com/index.html, accessed 20.09.2019.

[21] *Framework Usage Distribution in the Top 1 Million Sites*, https://trends.builtwith.com/framework\, accessed 20.09.2019.

[22] Rathore, A. *How to translate in Ionic 4—Globalization, Internationalization and Localization*, https://enappd.com/blog/how-to-translate-in-ionic-4-globalization internationalization-and-localization/11/ , accessed 20.09.2019.

[23] *Apache Cordova, globalization.getLocaleName*, https://cordova.apache.org/docs/en/2.5.0/cordova/globalization/globalization.getLocaleName.html, accessed 20.09.2019.

[24] *Mobile Operating System Market Share Worldwide*, August 2019, http://gs.statcounter.com/os-market-share/mobile/worldwide, accessed 20.09.2019.

[25] *App Translation service in Google Play Developer Console*, https://support.google.com/l10n/answer/6341304?hl=en, accessed 20.09.2019.

[26] *Passive localization JavaScript library*, https://github.com/eligrey/l10n.js/, accessed 20.09.2019.

[27] Lakó, C. On Internationalization (I18N), *Studia Universitatis Petru Maior. Philologia*, Vol. 19, pp. 151-160. Available at: https://www.researchgate.net/publication/293653270_ON_INTERNATIONALIZATION_I18N, accessed 20.09.2019.

[28] Sethi, I., V. Agarwal, S. Shukla. Mobile App Testing: Challenges, Strategy and Approaches. *International Journal of Computer Applications*, **No. 43** (vol.179), May 2018, pp. 16-22.

[29] Strzelecki, A., A *Framework for App Store Optimization*, https://www.researchgate.net/publication/333445038_A_Framework_for_App_Store_Optimization, accessed 20.09.2019.

Information about the authors:

Associate Professor Nikolay Kasakliev, PhD, Department "Computer Science", University of Plovdiv "Paisii Hilendarski", areas of scientific research: e-learning, information security, programming languages, mobile development, etc.

Associate Professor Elena Somova, PhD, Head of the Department "Computer Science", University of Plovdiv "Paisii Hilendarski", areas of scientific research: e-learning environments, educational content modelling, gamification, programming languages, etc.

Margarita Gocheva, PhD student, University of Plovdiv "Paisii Hilendarski", areas of scientific research: software for mobile devices, game-based learning, etc.

Manuscript received on 10 October 2019