Computers, Materials & Continua
DOI:10.32604/cmc.2021.013587
Article

Tech Science Press

Technology Provides Better Document Search Results on Slovak Legislation Webpage as Result of a Simulation of Webpage Performance Parameters

Peter Kvasnica[*]

Ministry of Justice of the Slovak Republic, Bratislava, 81311, Slovak Republic
[*]Corresponding Author: Peter Kvasnica. Email: peter1.kvasnica@gmail.com
Received: 12 August 2020; Accepted: 12 September 2020

Abstract: This article acquaints the public with the insights gained from conducting document searches in the Slovak public administration information system, when supported by knowledge of its management. Additionally, it discusses the advantages of simulating performance parameters and comparing the obtained results with the real parameters of the eZbierka (eCollection) legislation webpage. This comparison was based upon simulated results, obtained through the Gatling simulation tool, versus those obtained from measuring the properties of the public administration legislation webpage. Both sets of data (simulated and real), were generated via the the document search technologies in place on the eZbierka legislation webpage. The webpage provides users with binding laws and bylaws available in an electronically signed PDF file format. It is free open source. In order to simulate the accessing of documents on the webpage, the Gatling simulation tool was used. This tool simulated the activity, performed in the background of the information system, as a user attempted to read the data via the steps mentioned in the scenario. The settings of the simulated environment corresponded as much as possible to the hardware parameters and network infrastructure properties used for the operation of the respective information system. Based on this data, through load changing, we determined the number of users, the response time to queries, and their number; these parameters define the throughput of the server of the legislation webpage. The required parameter determination and performance of search technology operations are confirmed by a suitable hardware design and the webpage property parameter settings. We used the data from the eZbierka legislation webpage from its operational period of January 2016 to January 2019 for comparison, and analysed the relevant data to determine the parameter values of the legislation webpage of the slov-lex information system. The basic elements of the design solution include the technology used, the technology for searching the legislative documents with support of a searching tool, and a graphic database interface. By comparing the results, their dependencies, and proportionality, it is possible to ascertain the proper determination and appropriate applied search technology for selection of documents. Further, the graphic interface of the real web database was confirmed.

Keywords: Legislation webpage; simulation tool; search tool; performance testing

1 Introduction

The Government of the Slovak Republic approved the Strategy on Informatization of the Public Administration (hereinafter as the "Strategy") by resolution of the Government of the Slovak Republic No. 131/2008. The Strategy defined the vision, strategic goals, and direction of e-Government in the Slovak Republic until 2013, including its management structure, implementation plan, and framework of funding sources for the realisation of strategic goals. Another relevant source of guidance is a document from the Slovak Republic's Ministry of Finance in the field of public administration informatization on the national level, which makes reference to the National Concept of Public Administration informatization. This Concept is based on the Strategy and the Act on Information Systems of the Public Administration (hereinafter as the "ISPA") and it supports fulfillment of the government's program statement from August 2006, with an indicative time of implementation of until 2013, as per the Operational Program Informatization of Society [1]. A substantial part of this article deals with the comparison of simulation results to the results of the real Ministry of Justice of the Slovak Republic's (hereinafter as the "Ministry of Justice") legislation webpage.

Large and significant changes in webdesign development are difficult when attempting to fulfill the delivery requirements of effective client-server type applications, especially if those changes are additionally conditioned on a change of document search technology. Application complexity continues to grow exponentially, and business and economic expectations are extremely strict. The development of client-server applications for the webpage eZbierka places the emphasis on the external environment, and therefore must respect current requirements. The technological level is "only" a means for effective administrative performance, and is implemented in accordance with the defined competencies, tasks, and duties [2].

The eZbierka legislation webpage provides, in electronic form, non-binding documents obtained in accordance with the required standards [3]. The graphical user interface enables viewing authorized legal Acts and a consolidated version of the authorization. The eZbierka webpage was created in such a way that it is accessible and fully functional in all of the most commonly used web browsers of modern mobile devices, provided they meet the standards intended for the use of this category of devices.

The required performance of the legislation webpage server can be determined by monitoring the request processes and planned computer workload. The main aim of this article is to determine the workload of a webpage which depends on the designated search technology of its users to obtain documents, and model the workload of a computer determined through load prediction simulation. The secondary aim is to introduce the possibility of using the Gatling simulation tool, which enables the user to determine his or her behaviour through the set of web applications services contained in the user behaviour scenario. The tertiary aim is to introduce the results acquired from Log-on and General Users of the legislation webpage, implemented with the use of Ruby-on-rails technology for search for legislative documents with the support of the Apache Lucene Solr search tool and the Neo4j graphic database. In the proposed solution we used this concept of information system architecture design. See Reference [2].

2 Creation of Public Administration

The process of public administration informatization should result in an efficient electronic form of administration and increases in its efficiency, scalability, and use. Achieving such results significantly contributes to an increase of effective administration and provision of services to citizens and businesses. A contribution to efficiency means, in particular [2]:

- Increasing the process transparency, and thereby the response of public administration vis-a-vis the public
- Increasing the performance effectivity and cost reduction
- Reducing the time needed to address any agenda or issues, and likewise reducing the operators's administrative burden
- Improving information availability, etc.

The required quality of the information system's (hereinafter as the "IS") architectural design conditions profoundly affects the fulfillment of the prescribed services' functional requirements and subsequent achieving of the expected goals through the implemented processes. The quality of the design must support the fully required functionality, adaptability of modifications, and performance [4].

In the overall IS Project solution architecture design for a webpage with laws and bylaws, as an extension of the IS electronic collection and legislation project (hereinafter as the "IS SLOV-LEX"), a starting point was the modern principle of service oriented architecture (hereianfter as the "SOA") based on the TOGAF standard. Powerful open-source solutions were used that were scalable, easily maintained, and able to provide the planned consumer services in the required quality for IS use.

The basic solution is designed as a three-layer architecture, consisting of the presentation layer, the data layer, and a middle layer (middleware). The middleware itself consists of a layer managing access to imported documents, Fabasoft technology, communication interfaces, and technology used in the solution [5], Fig. 1. For the next analysis the data layer, which contains the reference data originally taken from other systems and source registers, is important. Reference data are necessary for the correct functioning of the components in the application layer [6].

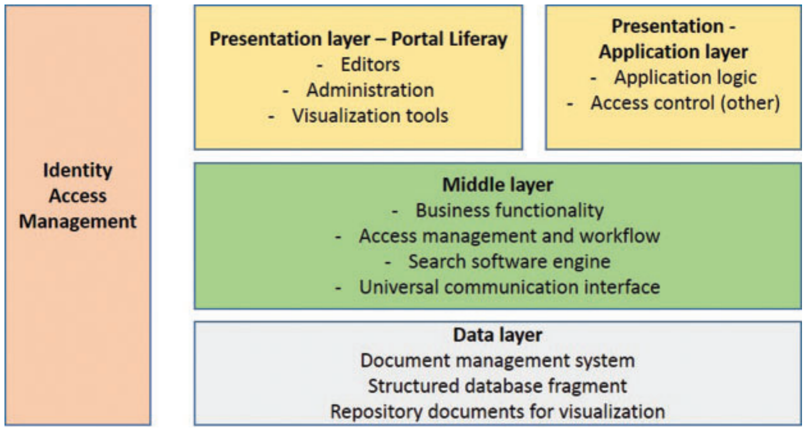

Identity Access Management
Presentation layer – Portal Liferay
- Editors
- Administration
- Visualization tools
Presentation - Application layer
- Application logic
- Access control (other)
Middle layer
- Business functionality
- Access management and workflow
- Search software engine
- Universal communication interface
Data layer
Document management system
Structured database fragment
Repository documents for visualization

Figure 1: Three-layer scheme of IS SLOV-LEX

The basic elements of the architecture of the entirety of the legislation webpage IS (SLOV-LEX) form:

- A single portal divided into logical parts, based on the Liferay Enterprise Portal technology [5]
- A presentation layer formed by editors and administration consoles
- Application logic, covering the complete business functionality and access management, based on a JBoss Application Platform and Fabasoft technologies for modules of workflow management
- A powerful search software core (engine) built on an Apache Solr solution
- A Document Management System—DMS, based on FabaSoft technology, as a primary repository of approved and imported binary documents
- A database repository of structured (fragmented) information based on Java Content Repository technology JSR 286 (hereinafter as the "JCR")
- A repository of pre-generated HTML and PDF versions of files for visualisation requirements on the portal for the filesystem.

Modeling the properties of the client-server technology enables repeated and rapid progress in developing the data layer application. For communication with external modules, the system uses the standards of REST and SOAP functions, in which the structured data is sent in JSON and XML formats. The Fabasoft system is used to integrate the back-end system requirements. To determine user behaviour and the related amount of documents for a search to be provided by the web application of data layer, the Gatling simulation tool was used.

3 Gatling Tool Features

Gatling is an open-source load and performance testing framework based on Scala, Akka, and Netty. The first stable release was published on January 13, 2012 [7]. In June 2016, Gatling officially presented Gatling FrontLine, Gatling's Enterprise Version, with additional features [8]. The software is able to identify the load that applications and infrastructure are ready to absorb.

3.1 Gatling Software

The software is designed as a load testing tool for analyzing and measuring the performance of a variety of services, with a focus on web applications. Gatling was mentioned as "a tool worth trying", with an emphasis on "the interesting premise of treating your performance tests as production code" [9].

The Gatling tool is appropriate for the performance testing of web applications. The Project's aims include creating high performance HTML reports and a developer Digital Subscriber Line (hereinafter "DSL"), or scenario records. The simulation file includes the different test scenarios, its parametrization, and the injection profiles. The Scenario consists of a series of requests where each sub-scenario within a simulation can have its own injection profile. Groups can be used as a subdivision of a scenario, as well as being a series of requests with a functional purpose. To simulate complex user behaviors, Gatling generates the appropriate requests in the system being tested. The injection profile is the number of virtual users injected into the system during the test, and the method in which they are injected [8].

Gatling also generates an HTML report at the end of each test. The report includes all information about active users over time, the time response of the distribution, and the percentiles, requests, and responses over time, etc.

3.2 Requirements for Use of Gatling Software

To use the performance testing offered by Gatling, a Java Developer Kit must be installed; Gatling requires the JDK8. The performance of the web application is determined by a test where the user-tester creates scenarios that represent actual user behavior. Here is an example of real user actions:

- A user arrives at the application
- The user opens one of the related models
- The user returns to the home page
- The user iterates through the pages
- The user creates a new model.

Gatling allows the user to record their actions on a web application and export them as a Gatling scenario. As such, when running simulations, the tester should act as a real user would, making real connections, and not immediately jump from one page to another without taking the time to read the content.

3.3 Running Gatling Software and Application Testing

In the case of this scaled IS, i.e., multiple environments operating on the same hardware, the functionality, availability, comfort, complexity, and reliability of the data obtained remotely from the user are all important. For such a comprehensive verification, various tests are used to focus on individual fields of an authenticated IS [10]. The aim of testing the performance parameters is to determine whether a defined or variable number of users are able to connect to the IS through a web browser, to specify the system requirements and, of course, to obtain the required data from the IS.

Testing is done by generating the load of the Gatling tool. The advantage of this instrument, compared to simpler alternatives (i.e., Apache Bench or Wrk), is the ability to test complex scripting scenarios. The Gatling tool used in testing simulates browsers, supports cookies, gzip, and keep-alive connections. Gatling is a highly capable load testing tool, which comes with excellent support of the HTTP protocol, making it the tool of choice for load testing any HTTP server. The core engine is perfectly able to implement other protocol support [11]. The context of testing, when developing load scenarios, should cover all of the common ways a user can navigate, though in practice it is necessary to simplify somewhat. The scenarios are code and resource efficient, on a DSL, and the scenarios are self-explanatory [12]. The most commonly-used performance metric is the response time on a throughput system. The paper examines the influence of a changing number of user web applications on the Slov-lex IS throughput.

4 Simulated Users: Quantity and Behavior

For the purposes of this chapter, “virtual user” means that user performing steps to search documents on the legislation webpage as would be performed by a real user in the IS. This includes those user steps on the real eZbierka webpage as realised in each user scenario. The Gatling scenario was used for recording search activities through the web application, simulating work on the data layer. This scenario was used to simulate reading the data using a web application. It includes client defined back-end operations for server authentifications and client requests to gain data. The set of these operations is referred to as an IS service.

Testing was performed through the Gatling tool to generate a simulated load. Several scenarios were simulated throughout the testing, but with respect to the scope of this article only one is described, as an example of the process of load testing when reading documents. This scenario consisted of reading of legislative documents located in the repository, reading them using the API web application as it performs the selection process in production without caching, waiting for selected documents, moving these documents, and other activities users are authorized to perform on the eZbierka legislation webpage. The eZbierka webpage allows access to two distinct types of users-those with log-on priveledges (Log-on Users) and those without (General Users). The distinction between the two relates to access privileges. Public administration workers require functionality that permits adding details and comments to legislative documents and processing metadata, etc.; granting this access requires that such users be logged on.

4.1 Simulated Server Hardware Parameters

As an example, testing of the electronic collection of laws of the eZbierka legislation webpage by users (both Log-on and General) was chosen. This activity is the most common when searching for documents, and thus has the biggest influence on the stability and performance of the IS. In these tests, (the definition of which can be found in [2]), the load on the individual system pages was simulated by gradually adding users up to the final limit of 5000 General Users or 1200 Log-on Users. These users requested access to the data stored on the eZbierka legislation webpage through a web application, which they ran on the server architecture. The test was performed paralelly using three testing servers with the following parameters:

- User settings of an application
- One open user way of reading data
- Ubuntu Linux 14.04 LTS
- Java(TM) SE Runtime Environment (build 1.8.0_25-b17), 64 bit
- Procesor Intel, Core i7-4790
- Memory RAM, 32 GB
- Access to internet: 1 Gbit of downloads, 1 Gbit of uploads
- Gatling tool version 2.2.0, to simulate user behaviour—load of legislation webpage.

4.2 Simulation of Legislation Webpage of Public Administration Users

In these tests, the load on individual pages was simulated by gradually adding virtual users up to the final limit of 5000 General Users or 1200 Log-on Users. These users asked for access to the data stored in the eZbierka IS through a web application, which works with a data layer according to operations specified in the scenario. The following rules were used to evaluate whether the system was behaving correctly:

- The reply from the server is not HTTP 404, 500, 301 (redirection to the static part) or 302 (temporary redirection to the static part)
- No timeout reply
- No disconnected TCP connection reply [13].

The HTTPS protocol is designed to ensure communication between the client and the server through its functions; the GET function is used to obtain the data from the source (server) and the POST function to send data to the source (server) and update it; for more information see [2]. The scenario consists of the following steps: From client's authentication (in the background) to the server, from reading the laws and bylaws from the data repository, reading through API corresponding to the selection during production and ensures prevention of caching, from waiting for selected documents, from transfer of these documents, etc.

4.2.1 Virtual General User Web Testing

The increase of users was gradual and proportional. At 0 s there were 0 users in the system, increasing to 5000 users in the system at 15 minutes. See Fig. 2 below. Each user performed activities which were included in the General User simulation scenario.

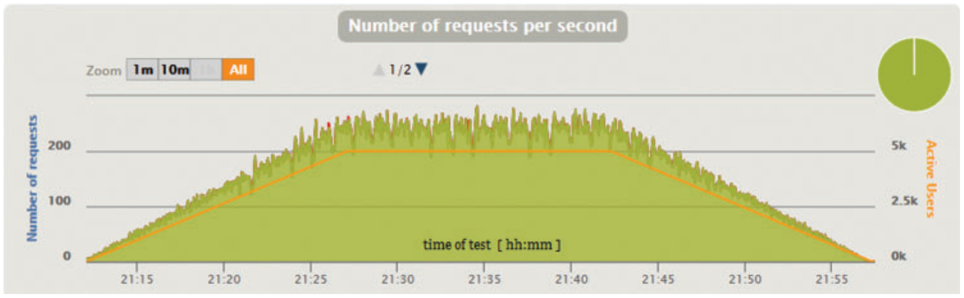


Figure 2: Number of requests regarding data from the server and virtual General Users

Throughout the 45-min loop, according to the scenario, each activity was performed by a virtual user (hypertext transfer protocol—http GET and POST functions). During the maximum load of 5000 virtual General Users, the system generated an average of 70 requests per second to access data. These requests meant calling a web application to read data from the data layer. During the test, 439,139 requests were sent to the server, to which the server responded within 3,190 ms in most of cases. At the 95th percentile, the request response time was 3,010 ms (i.e., 95% of requests were processed within 3,010 ms).

4.2.2 Web Testing by Virtual Log-on Users

The intention was to check whether the server could simultaneously serve a larger number of Log-on Users when reading the data according to the scenario [10]. The increase of Log-on Users was gradual and proportional. At 0 s there were 0 users in the system, and at the 10 min mark there were 1200 users in the system, an increase of an additional 2 users per second. After each performed action, according to the simulation scenario, the user waited the respective time, the proporsionality of the results were the same as in the test for General Users when reading documents. See Fig. 3. Public administration workers require functionality with the option to add details to legislation documents, to add comments to them, to process metadata, etc. The users must be logged on for these tasks, i.e., they must be signed on to the webpage IS.

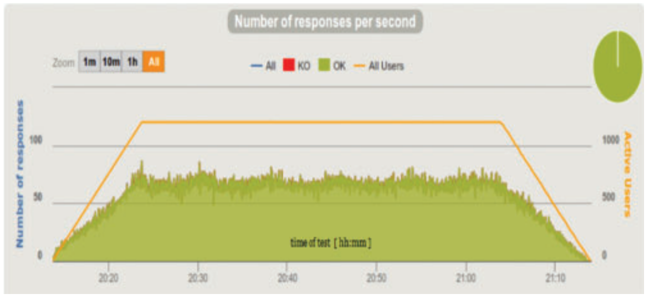


Figure 3: The number of requests regarding data and virtual Log-on Users of the system simulated in time

Throughout the 60-minute loop, each user performed the same actions (by calling the GET and POST functions). During the simulation, there were 204,248 requests sent to the server regarding data, to which the server responded within 600 ms most of the time. The average time to process the request for data from the web server was 820 ms. See Fig. 3. At the 95th percentile, the request response time was at the level of 750 ms (95% of the requests were processed within 750 ms).

5 The Real Number of User Searching and Filing Requests to the Legislation Web Page

The proposed system artifacts solution is in accordance with the best practices of SOA methodics, according to the reference architecture. This approach allowed a focus on the expected practical use of the information system by multiple users, simultaneously. A system so designed enables the paralel use of services intended to provide information from the legislation webpage eZbierka through the data layer. This required feature was reached by implementation of the Apache Lucene Solr searching tool technology and Neo4j graphic interference of the database. The result of the architecture design, as per the abovementioned rules, is that the principles of the SOA and TOGAF standards in the system meet the following requirements [14]:

- Clarity
- Robustness
- Completness
- Consistency
- Stability.

The system design is realised in accordance with the expected modern SOA architecture, which forms the standard for all the current and future solutions of information technologies in various fields. At the same time, in realisation of this project, the following principles and technology were applied [15]:

- Security at the network level by separating the environments and modules into separate zones
- Security at the level of operation systems by implementing internal firewalls
- Maximum possible virtualisation of the computing power.

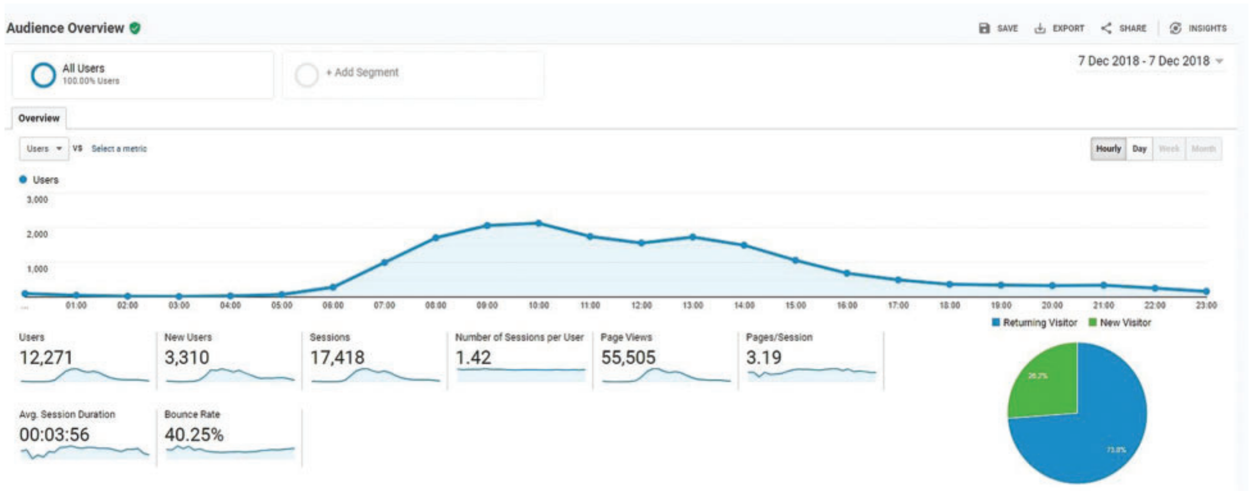
The basic idea is to use modern, open source resources and solutions which are sufficiently reliable. When selecting components, as emphasis was placed on how the individual components were able to mutually integrate and complement each other, and whether a reference solution existed where the respective integrations had been used. The key question is the integration of those components, into which it is possible to add further applications and modules in the form of portlets, meeting the JSR-286 standard, or other standards which, in the meantime, became industry standards [14]. We used technologies for searching legislative documents (Apache Lucene Solr search tool and graphic database Neo4j) in order to increase the performance of the webportal. The Ruby-on-rails technology enables creation of web applications more easily and quickly. This allows for easier code writing if more than one language is used in the implementation concept. These features allow the application to work faster when searching legislation, when compared to the expected results from the Gatling tool simulation.

The integration of the application modules based on the JSR-286 standard had the main influence on the performance of the web application for reading documents of the web portal, and the number of users (both Log-on and General) of the IS. This standard consists of the partial tree structure and describes the features available in Java Portlet. The web portal eZbierka publishes only adopted (consolidated) versions of the laws and bylaws, which were signed before publication [13].

The data was obtained from the measured data of the legislation webpage during the period from January 2016 to January 2019 through analytical tools [16]. Data from this period was compared with the previously mentioned results from Gatling simulation tool. In the implementation, where it was appropriate, Ruby-on-rails technology was used, allowing easier and faster development of the web applications. At the same time, in the solution, the Apache Lucene Solr and Neo4j graphic interference were used. The legislation webpage eZbierka makes available the adopted laws and bylaws in pdf format, i.e., these data are consolidated, and required metadata is added by Log-on Users.

5.1 *General User Results*

Another requirement was to dimension the number of users in such a way that the number of user requests for data should be covered by the performance of the server. On average, this number ranged from hundreds to more than 10 thousand users throughout the day. See Fig. 4. As such, comparison of these results with dependences shown in Fig. 2 is possible. During the working day, there were 12,271 General Users to the server, with a response time of 1,920 ms in most cases, and an average response time of about 1,810 ms (95% of requests were processed within 1,810 ms).



Users 12,271

New Users 3,310

Sessions 17,418

Number of Sessions per User 1.42

Page Views 55,505

Pages/Session 3.19

Avg. Session Duration 00:03:56

Bounce Rate 40.25%

Figure 4: The number of General Users of the legislation webpage during the working day [16]

The data obtained from the legislation webpage at this time are shown in Fig. 5. From the comparison with Fig. 2 it can be seen that a smaller amount of requests for data from the webpage were recorded with technology JSR-286 during the current operations, when compared to the number of requests determined from the simulation.

The value of requests for concurrent reading of data, in the following figure, ranges from 60 to 400 current requests. The real number of recorded requests was different from the data obtained due to the concurrent increase of the number of users, which does not occur in practice. The web application, the more detailed Apache Lucene Solr searching tool, and its implementation using

Ruby-on-rails technology, significantly contributed to the acceleration of providing documents when reading.

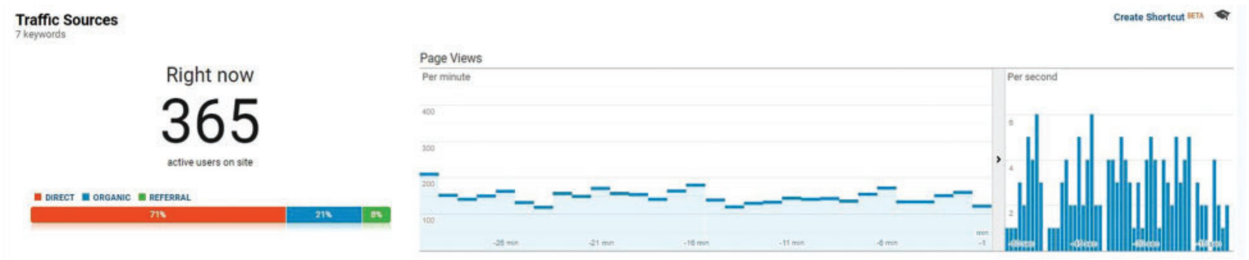
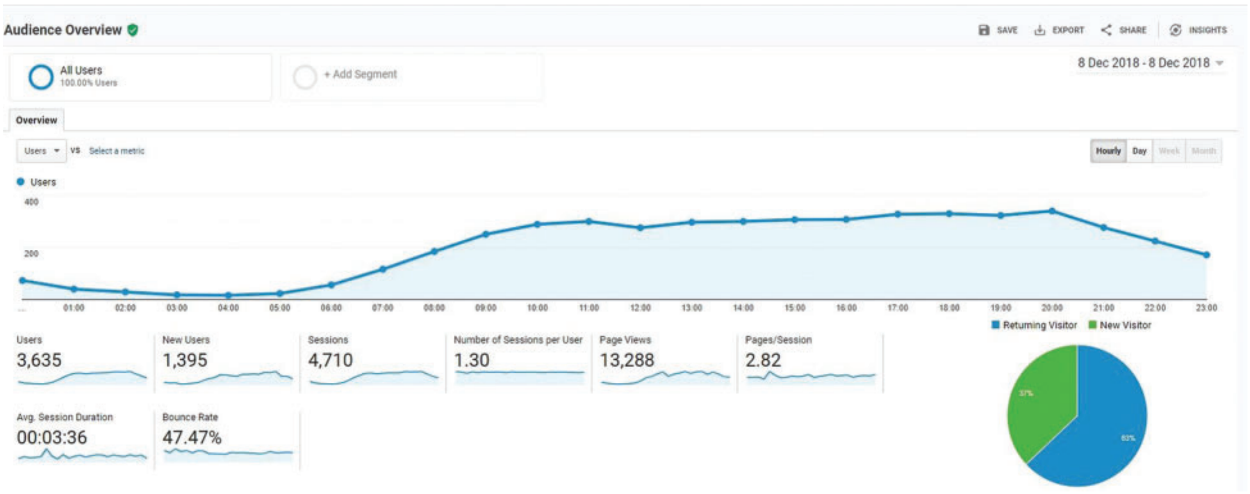


Figure 5: The number of current requests to read the data of the legislation webpage during the monitored time [16]

5.2 *Log-on User Results*

The number of users was determined in such a way that users with requests for data were monitored simultaneously, with more than 300 users requesting data at the same time. See Fig. 6. The probability of reading the legislation webpage can be compared to the dependencies from Fig. 3. During the working day, there were 3,635 Log-on Users accessing the server, with a response time of up to 545 ms, the average time of response was around 500 ms (95% of requests were served within 500 ms). From the comparison of the data on these results, it is clear that the current number of users requesting data is lower than the number of users in the simulation tool. The simulated number of users, with their requests, was generated on the basis of a steady increase in the number of users in the test, which is very unusual in practice. The substantial difference in acceleration of document provision, when reading the web application for Log-on Users, is due to the use of the Neo4j graphic database interference and its implementation using Ruby-on-rails technology.



Users 3,635
New Users 1,395
Sessions 4,710
Number of Sessions per User 1.30
Page Views 13,288
Pages/Session 2.82
Avg. Session Duration 00:03:36
Bounce Rate 47.47%

Figure 6: Number of Log-on Users of the legislation webpage during the working day [16]

Numerical differences in methodology revealed the possibility of improving the application in the number of processed user requests for data. The data, according to Fig. 7 below can be used to determine the technical work device performance of the legislation webpage eZbierka. The number of requests to read data is shown in the following figure, the number is from 25 to 250 current requests. The simulation results may be affected by the priority of an application that simulates reading data from the server, which can help retrieve recorded data faster.

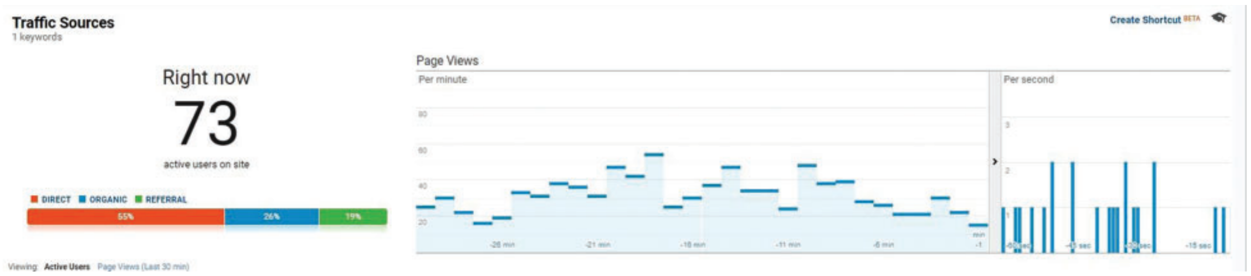


Figure 7: Number of requests to the legislation webpage in the monitored time [16]

6 Conclusion

The public expects the government, as creator and operator of the public administration's web information services, to reach the perfect level of provided public services, especially concerning its quality, accuracy, and accessibility. The number of documents searched by users, their accessibility, quality of services, and preparation, are key factors for further strategic development. This access increases the use management transparency of the webpage, it reduces the administrative burden for users, and it creates comprehensive support for changes to the public administration's legislation webpage.

With respect to the results of the performance tests, these requirements were met when implementing the IS of the legislation webpage www.slov-lex.sk. The results were obtained by simulating the number of users and proprieties through scenarios which, by their methodology, emulate the behaviour of real users. The users call the services to obtain data from the webpage of public administration, their usage determines the response time and the number of answers, and from these parameters they test the extent of the performance of the legislation webpage server. These results were compared with the results from the web application implemented using Ruby-on-rails technology for searching legislation documents with the support of the Apache Lucene Solr searching tool and the Neo4j graphic database interference. Different results were seen when compared to the real data, because the selected data on the eZbierka legislation webpage was measured which is affected by the technology it used. In case the webpage offers longer time responses compared to the detected ones, or the number of real users increases significantly, it will be necessary to change its hardware to strenghten its performance, i.e., to increase the throughput or performance of the server.

The system is built in such a way that it is expressly modular, what enables rapid change and improvement. It is open to external systems and uses open components (open source) in the implementation. Fulfilment of the respective determined conditions is supported by high quality and sophisticated design of the architecture of the solution. The throughput of the system enables a dimensioned number of users to make use of the data search services and meets the performance requirement, together with all the available data on the webpage www.slov-lex.sk. The Ruby-on-rails technology, the Apache Lucene Solr search engine, and Neo4j graphic database fully support the required functionality, especially their technical parameters and requests for performance. The use of the abovementioned technology enabled achieving better results than expected according to the results from the simulation tool. This knowledge allows the conclusion that, after the implementation of even newer technology into the information system, better results will be seen, which will lead to greater satisfaction of the users of the legislation webpage, and this is essential.

Funding Statement: The Author would like to thank the Operational Program Informatization of Society No. 2110002 for use of its data and OPIS-2011/1.1/36-NP for the requesting the research.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] OPIS, “Operational program informatization of society,” 2015. [Online]. Available: http://www.informatizacia.sk/opis/598s.

[2] P. Kvasnica and M. Zimany, “Web availability by simulation of selected parameters of power loads,” in *IEEE Proc. of the 13th Int. Conf. on Informatics*, Slovak Republic, pp. 161–167, 2015.

[3] Slovak Republic Legislation, “Informatization of Society,” 2015. [Online]. Available: http://www.informatizacia.sk/Legislativa/Legislativa.

[4] J. Hanzlíková, *Webdesign*, 1st ed., Brno: Computer Press, pp. 19–46, 2004.

[5] Apache Solr, Solr Features, “APACHE SOLRTM 5.2.1,” 2015. [Online]. Available: http://lucene.apache.org/solr/features.html.

[6] J. Nielsen, *Web.Design*, 1st ed., vol. 2. Indianapolis: Pearson Education, Inc., pp. 28–105, 2000.

[7] Gatling Reached, “Gatling has reached 800,000 downloads,” 2017. [Online]. Available: https://gatling.io/docs/current/quickstart/.

[8] Soirée de présentation Gatling FrontLine, Meetup, Gatling paris user group,” 2017. [Online]. Available: https://www.meetup.com/fr-FR/Gatling-User-Group-Paris/events/229956473/.

[9] R. Tolledo, “Gatling: Take your performance tests to the next level. ThoughtWorks (12 May 2014),” 2017. [Online]. Available: https://www.thoughtworks.com/insights/blog/gatling-take-your-performance-tests-next-level.

[10] P. Farrell-Vinay, *Manage Software Testing*, 1st ed., vol. 16. Broken Sound Parkway New York: Auerbach Publications, Taylor & Francis Group, 2008.

[11] Netty Akka, “Gatling is an open-source load testing framework based on scala,” 2016. [Online]. Available: http://gatling.io#/docs.

[12] M. Li, N. Cao, S. Zou and W. Lou, “Privacy-preserving personal profile matching in mobile social networks,” in *Proc. IEEE INFOCOM*, Shanghai, China, pp. 2435–2443, 2011.

[13] Project SLOV-LEX, “Description electronic collection of SLOV-LEX,” 2015. [Online]. Available: http://www.informatizacia.sk/OPIS.

[14] OpenGroup Standard SOA, “SOA reference architecture technical standard,” 2015. [Online]. Available: http://www.opengroup.org/soa/source-book/soa_refarch/layers.htm.

[15] H. Schildt, *Java 2 Příručka programátora*, 1st ed., vol. 12. SoftPress, Czech: Osborne, McGraw-Hill Inc., 2001.

[16] Technical Information MoJ SR, “Skratky, prehľad, aktívni používatelia, rýchlosť stránok—časovanie, kanály,” 2019. [Online]. Available: https://analytics.google.com/analytics/web/#report/defaultid/a60613354w95105056p99131185.