ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 7, Num. 2, 2022

CLIENT-SERVER LIBRARY INDEX AUTOMATION SYSTEM

***Nazar Repianskyi*[1], *Taras Rak*[2]**

[1,2]*Lviv Polytechnic National University, 12, Bandera Str, Lviv, 79013, Ukraine.*
[2]*IT STEP University, 83a, Zamarstynivska Str, Lviv, 79019, Ukraine*
Authors' e-mails*: nazar.repianskyi.mkiks.2021@lpnu.ua; taras.y.rak@lpnu.ua;*

https://doi.org/10.23939/acps2022.__.___
Submitted on 26.09.2022

***Abstract*: The goal of the work is to develop a client-server software system for library index automation, which consists of the user interface, presented as a web page, back-end server layer, and database. The problem of developing a library index automation system has been considered. The developed structure of the library index automation system has been presented. The structure of the library index automation system database has been proposed. The features of designing the user interface of the library index automation system have been considered. The general algorithm of the library index automation system and the system's class diagram have been presented. To make sure that the developed system is secure and stable development testing as well as stress testing have been performed.**

***Index Terms*: client-server architecture, software system, library, automation, database, Java, Spring, Online public access catalog.**

I. INTRODUCTION

Data warehouses are a very important pillar of society because they are the ones that can continue humanity's progress, despite the significant increase in the information on which it builds. We all know the word library, although it meant a storehouse of books at the time of its creation, actually it has an informational meaning completely analogous to data storage.

Although traditional libraries are slowly transforming into virtual and digital libraries with the help of ICT and the internet revolution [1]. Now, as in the past, libraries store information in different forms and on different media. Such distribution is necessary to ensure the integrity of information, a violation of information storage in only one group which greatly increases the possibility of its loss [6].

Thus, it can be considered proven that the opinion about the rudimentary nature of libraries, which has begun to take shape in modern society, is wrong. Libraries, as data repositories, are and will be necessary regardless of time, and they do not depend on the forms of information carriers used in one or another era.

As of today, a lot of libraries still use paper-based indexes, which have quite a number of negatives, as they take a lot of space, age badly, are difficult to update, are slow to search info, and the most important – must be physically managed by humans, what leaves a huge security flaw and gives a lot of room for mistakes.

And to avoid these negatives, to reduce the amount of work and make indexes more user-friendly and faster, as well as improve security and stability, and help in library promotion, it was planned to create a client-server library index automation system, which also includes the functionality of OPAC (Online public access catalog).

This paper presents the results of the development of a such system that provides a user interface in the shape of a webpage allowing the automation of different numerous functions for people with different access levels to the library.

It helps everyone to look through the catalog, allows authorized users to look at the list of books he is yet to return, orders on borrowing books and requests on returning them, and allows librarians to update, delete and add data in the index, as well as track all the info on pending and finished orders for borrowing and requests for returning books, and info on books held by any user, allows managers to accept or decline orders and requests, allows a director to look over all the processes of the library, helps manage employees account data, and allows to register or delete accounts. And of course, it allows every user to manage their account data.

The system also allows flexible search through the catalog by filtering books by words their name contains, filtering them by genres, or filtering by both criteria.

All of that functionality is distributed between access levels and protected not only in the webpage but also in the webserver layer. The system provides encryption of user data and can store that encrypted data on a local or remote database server. It also stores user sessions in the database, what frees the user from the need of putting his credentials every time he logins to the system, reduces to amount of unnecessary actions, and improves the user experience.

Library [1] is a cultural and educational institution that collects printed and manuscript materials, conducts their processing, and display in catalogs, organizes their appropriate storage, preservation, and service of their readers, as well as provides promotion and distribution of literary works.

Education is the pillar of development of society and library is its integral part devoted towards development of intellectual society [2].

A card index is a system of cards with accounting, references, and other information. It is also a box or cabinet for storing cards. Each card in the index is an information unit that provides information about any object that is stored to ease searching for that object by certain tags. It is mandatorily sorted by logical criteria: according to the alphabet, date, etc.

The card catalog was a familiar sight to library users for generations, but it has been effectively replaced by the online public access catalog (OPAC). Some still refer to the online catalog as a “card catalog”.

As a typical database, Online-Public-Access-Catalogue (OPAC) refers to a database with a comprehensive information system that integrates all information that libraries can provide to meet the broad information needs of users [3].

In Client-Server architecture processing and storage of the data is done by the server part of the system, while the client part is responsible for the display features and provides an interface to a user. This way data is always safe and secure and the user interface is independent of the server part.

II. PROBLEM STATEMENT

Libraries are systematically engaged in collecting, storing, popularizing, and issuing printed works to readers, as well as information and bibliographic work. They are publicly available sources of knowledge and the main basis for self-education. The main areas of work of any library are stocking and organization of the book fund, as well as reader service.

Library activity is related to taking into account a large number of operations, many books and readers seriously slow down the work of librarians. The difficulty of finding the right book in the catalog takes a long time and depends entirely on the competence of the library staff.

Library automation implements the concept of information and communications technologies (ICT) which are designed to address the problem of manual processing of material in the library in order to save the librarians time and energy [4].

The automation of the library index would improve its stability and security, in addition to speeding up the working process including search and update.

The usage of client-server architecture is mandatory because library workers have to access the same set of data from different workstations, and that set of data should stay safe and secure, also it would be preferred to know its physical location. Another advantage of this architecture is that anything that has a browser can be a client. So, users can access the system from PC, notebooks, tablets, smartphones, smartwatches, etc. Storing data in the database also gives a lot of tools for data processing, helps to keep data structured, makes it easy to export, and improves the total productivity of data processing. Using the website as an interface helps to keep the system simple and promotes the library on the Internet. It helps to keep high productivity with the lowest system requirements for client hardware. And it also improves the security, stability, and fault tolerance of the system.

Now, if you stick to the principle of platform independence, and take a look at website server operation system statistics (Fig. 1) taken from the W3Tech website [5], it can be noticed, that it is worth making server side of the system platform-independent as well. According to w3tech.com, they analyze HTML headers, elements, and specific tags, as well as responses to specific requests, together with JavaScript and CSS code and DNS records. Thus, the statistics they give can be considered trustworthy.

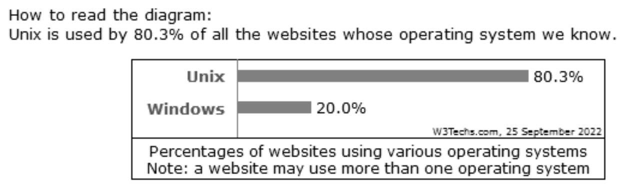


Fig. 1. OS by website usage statistics from the W3Tech website

While thinking about the modern programming languages suitable for the development of server-client applications, Java is the one, that immediately comes to mind, and also the one that is most commonly used nowadays.

Java can run the same program on different operating systems because Java applications are compiled to bytecode that can run by JWM regardless of the architecture of the systems it runs on. Other advantages of Java are exception and error handling and leverage with libraries. Also, it’s highly scalable and the code is structured and easy to navigate and understand.

While JVM is the main advantage of Java, it also can be viewed as the main disadvantage. Java programs take a long time to compile and use more resources than some other programming languages. Still, those flaws mean nothing from the client-server perspective, since all the processing is handled by the server, and clients get no hardware requirements. And taking breaks for technical maintenance is general practice nowadays, so you shouldn’t worry about compilation time.

The most common way to develop server applications is the usage of Java’s Spring Framework. In today's reality, the communication of client and server parts plays a key role in the execution of software [7], and Spring helps to maintain flexible and secure connections between layers. It is built with Inversion of Control (IoC) and Dependency Injection (DI) features, which provide the foundation for a wide-ranging set of features and functionality. The main features of Spring Framework are that it manages all architectural layers used in the development of web applications, and it allows to freely link dependencies and easily test them. Also, it supports declarative programming. It eliminates the need for manual creation of factory and singleton classes while supporting various configuration methods and providing a middleware-level service.

Library automation usually covers all library housekeeping functions such as acquisition cataloging circulation and serial control [8]. The automation system should contain the basic functionality of looking through the catalog, as well as user authorization and registration. The system needs different types of accounts for employees to manage the card index. Employees should be able to manage the catalog and track the state of operations inside the library, and clients also should be able to track the state of the operations related to them. All users should be able to manage their accounts. While director should be able to manage all of the users' accounts and register new employees, in addition to managing all the operations inside the system. Finally, the library index automation system should also implement search and sort functions.

There are also some other convenient management functions that should be provided to improve user experience and simplify interactions with the system for clients and employees.

The best way to provide access to numerous individual components of the system is to use a navigation panel, which can be different for various access levels.

The following user access levels should be provided in the client-server automation system of the library index: unauthorized user, the client, librarian, manager, and director.

Several security measures should be used in this client-server library index automation system. When receiving HTTP requests from the browser, which relate to the functions of a certain access level, it should be checked whether the user has the required level. Additionally, users with different access levels should have a different set of URLs in the navigation bar. Navigating or sending HTML requests to URLs outside of this set (that is, pages that are not accessible at the user's current access level) should result in a page-not-accessible error in the user's browser. And all URLs that are not included in the user's set, have to redirect the user to the login page. There should be access to the registration page from the login page. When registering users, from this page, they should be given the access level "Client".

In addition to the above functions, it is possible to separately store the user's session in the database, which allows him to log in less often. Functions for sorting books in alphabetical order by name and sorting orders and requests by time, the fact of confirmation and the fact of completion of orders/requests, as well as, the breakdown of all the data that is issued in tables into pages is necessary to make the system faster and more user-friendly.

III. PURPOSE OF THE WORK

The purpose of this work is to develop a client-server system that could automate the library card index, which would be able to work with a large amount of data and would ensure their security, both in terms of their encryption and in terms of their physical storage on the server, which can also be located inside the library. The system that would be more convenient to use for employees than a paper card file, would greatly reduce the amount of daily routine work, that they have to do, and would make it possible to significantly speed up the performance of library functions and make them more accessible.

The system's primary function is to maintain a list of books with a display of available and borrowed books. Maintaining a list involves adding new books, deleting books, and editing book information. In addition, it is necessary to display books owned by a certain user. Also, the system requires user authorization and registration. Since the client-server automation system must also include the borrowing of books by the user, it is necessary to enter the minimum required user access levels, such as employee and customer.

The resulting program should include all the above functions and correspond to the principles of the client-server architecture as well as include several additional convenient management functions that help to automate library routine and simplify interactions with the system.

IV. THE STRUCTURE AND WORK PRINCIPLES OF THE SOFTWARE SYSTEM

The general structure of the developed client-server library index automation system (Fig. 2) shows different processing layers of the developed application, along with the way of interaction between them.

The user interacts with the client-server automation system of the library index using the interface provided by the client part of the system.

The client part exchanges data with the server part of the system using HTTP requests and displays the HTML pages received from the server part with the attributes attached to them.

Accordingly, the server part processes the received requests with the help of the program, and the program itself already operates the objects that are created based on the data received from the HTTP requests. The program processes the data according to the specified logic, and, if necessary, accesses the database using SQL queries. Data obtained from the database are also transformed into objects that are manipulated by the program. So, the back-end program can put these objects into attributes, and send HTML pages with these attributes to the client side of the system.

The processing of the data received from the client part of the client-server automation system of the library index is not handled by the entire program, but by classes designated for this purpose, called Controllers. Controller methods are called depending on the URL of HTTP requests and the access level of the user who sent them. Thus, although the controllers are divided into different classes, as part of the separation of the logic of the parts of the program, the use of their methods does not depend on the part of the program where they are written, but only on the URL and the level of access that is set for these methods. It is also highlighted that only the controllers are responsible for the creation and initialization of objects received from HTTP requests, as well as for the preparation of HTML pages for the server part of the client-server automation system of the library catalog.

To process the received data, or obtain data from the system database, the controller uses the methods of the corresponding service. The service contains all the logic of the application, it processes the data and decides when, how, and what data should be obtained, deleted, or changed. The service also prepares the data that will be used by the controller. However, the service does not have a direct right to receive any access to the system database. To obtain such access, the service refers to the methods of the corresponding repository.

The repository does not contain logic and is intended for direct access to the database. The Repository in the Java data type system is an interface, which means that the repository itself does not contain any method implementation code, only its headers. These headers, as well as their annotations, are used by Spring to generate SQL queries, which are used to access the database of the client-server automation system of the library index. Accordingly, the repository serves only for access to the database of the system and performs the following: the addition, modification, deletion, and selection of data from the database, and transfer of this data to the appropriate service.

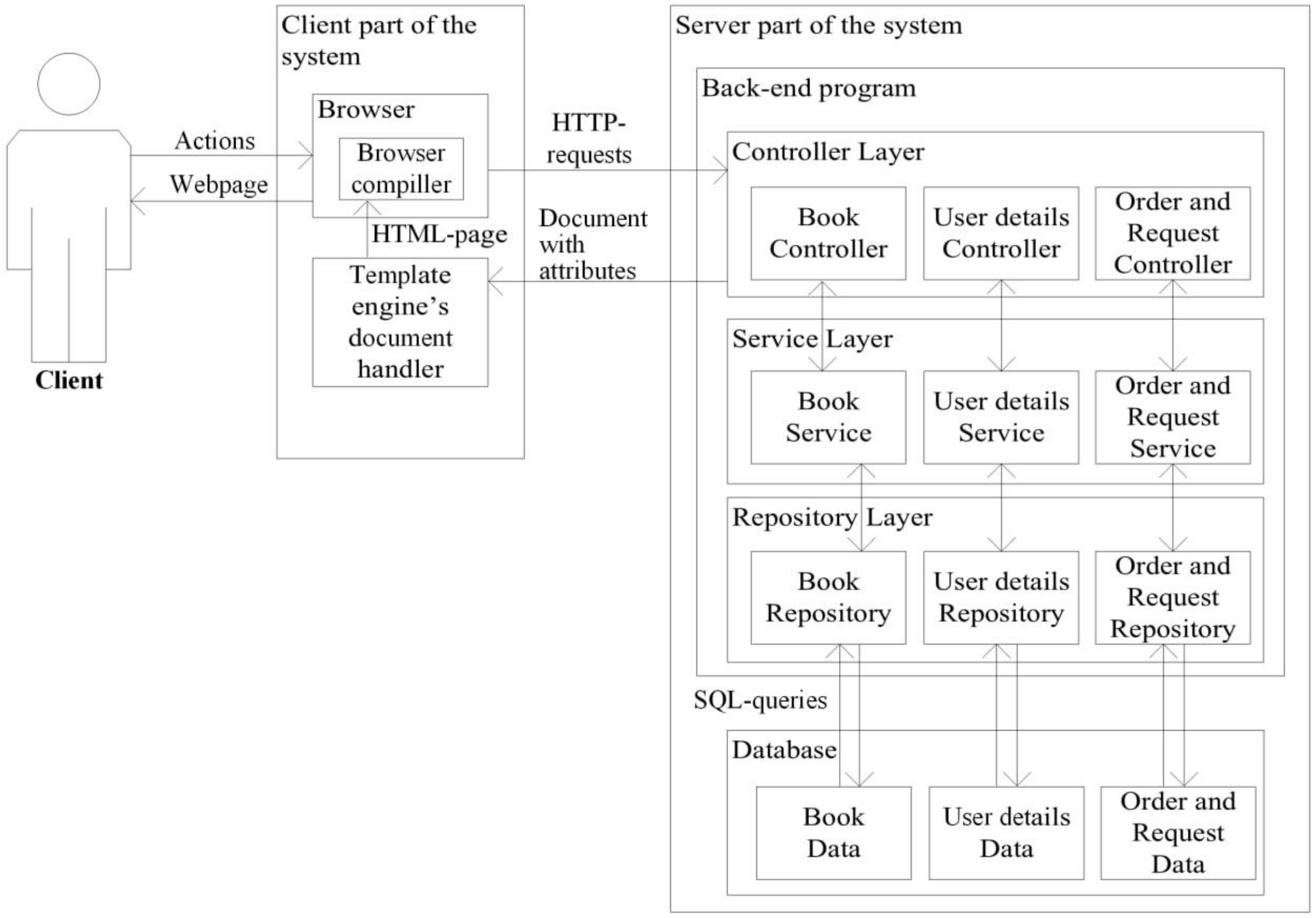


Fig. 2. The outline of the general structure of the client-server library index automation system

V. ALGORITHM OF OPERATION OF CLIENT-SERVER AUTOMATION SYSTEM

The general workflow of the client-server system of library file automation can be described as follows:

1) requests from the user's browser are accepted by controllers;

2) controllers validate data and return the specified template or transfer data to the services.

3) services perform all the necessary operations on the data and refer to the repositories to obtain access to the database;

4) repositories form requests to the database based on the data received from the services and the name of the repository function used by the service;

5) repositories return the execution result of the method used by the service if that is provided by the signature of this method;

6) service returns the data received from the repository to the controller, if, again, this is provided by the method signature;

7) controller places the data received from the service into the HTML attributes of the page, if called method returns data, and returns an HTML page or a redirect to a specific URL.

The second point in this chain is optional. The presence of validation rules, the rules themselves, and actions in case of their non-fulfillment are completely specified by the developer. And not all methods of controllers contain them. However, the purpose of creating this chain was to display the functionality of the classes, so the main thing in it is the order of possible actions.

Different classes of the developed system (Fig.3) belong to different layers of the system. The first word in the class name indicates which entity it works with, and the second one indicates the layer class works in. If the second word is absent, the class belongs to data entities, which are used as objects Spring operates with while exchanging data with the database. Classes that start with "Config" contains the framework configuration. Classes “PasswordEncoder” and “MailSender” are services that are used inside other services to encode passwords and initiate mail sending process.

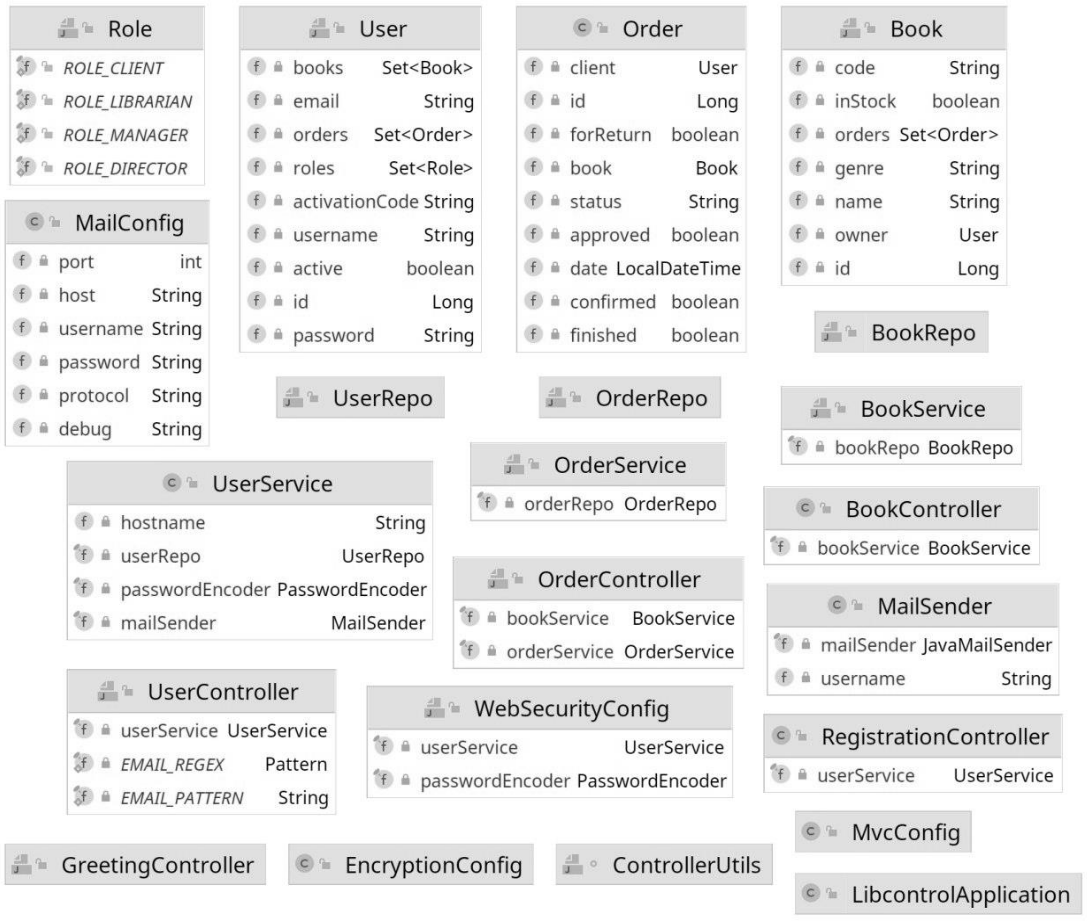


Fig. 3. Simplified UML Class diagram of client-server library index automation system

VI. THE STRUCTURE OF THE DATABASE OF THE LIBRARY INDEX AUTOMATION SYSTEM

The database of the client-server system of automation of the library index is implemented on PostgreSQL [9]. PostgreSQL is an open-source DBMS. This means that the system is free, easy to learn and integrate, additionally, it is scalable and open-source. While PostgreSQL is free, it is more productive and functional than many paid competitors. Additionally, PostgreSQL is supported on all major operating systems: Linux, Unix, Windows, and Mac.

The software module for Flyway database migrations was also used, which provides the functionality of modifying the database structure during the development of the system. Also, Flyway initializes the initial database structure, in particular: creates tables, creates users by default, encrypts their passwords, etc.

The database structure implements all the necessary tables, fields, and relationships between tables for storing platform data. The main part of the database structure (Fig. 4) contains the following tables:

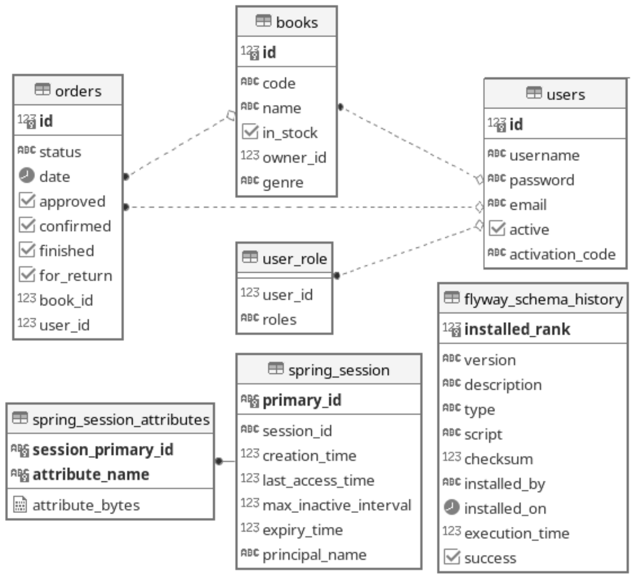


Fig. 4. The outline of the database structure of the client-server library index automation system

The "flyway_schema_history" table is created automatically by the Flyway software module and contains the history of database migrations. This table is used exclusively by this software module and is necessary for its correct operation.

The "spring_session" and "spring_session_attributes" tables are created automatically by the Spring framework. They contain information about user sessions and are also used and configured directly by the framework. Saving user sessions allows users to avoid logging into the library file automation system each time they use it. As it is seen from ER, the relationship between these tables is One to One. The information in these tables is also not directly processed by the developer.

The "users" table contains information about users. It is connected by a "One-to-many" relationship with the tables "user_role", "books", and "orders".

The "user_role" table serves to store user roles (access levels). It contains two fields: roles, which is a list of user roles, and user_id, which is used to establish a Many-to-One relationship with the "users" table;

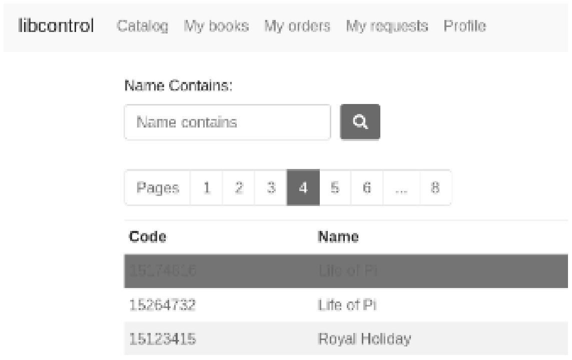
The "books" table serves to store information about books in the system. It is connected by "One-to-many" relationships with the "users" table and "Many-to-one" relationships with the "orders" table;

The orders table serves to store information about requests for receipt and requests for the return of books by the user. It is connected by "One-to-many" relationships with the "books" and "users" tables.

VII. THE USER INTERFACE OF THE LIBRARY INDEX AUTOMATION SYSTEM

The client part of the client-server automation system of the library index is implemented in the form of a web application. Accordingly, the user interface looks like a multi-page website. In the development of the user interface, the Java programming language, the Spring framework, as well as the FreeMarker template engine, the extensible XML markup language, and the Hibernate framework were used. Four independent levels of access are implemented in the user interface. The non-registered user interface contains only a page with a list of all books. In addition to that, the librarian interface allows to make changes to the book catalog. The user interface of the system (Fig. 5) also contains the pages of his orders and requests, as well as the page of his books. The manager interface contains all the same pages as the user interface. But it allows to manage requests and orders, and also displays them all. In addition to these pages, the director's interface allows adding of new employees to the system. In addition to these pages, each user can edit their account data.

Catalog Page (Fig. 5) consists of a navigation bar, input field, and button for filtering data, a table to display info about a book (such as code and name), and a row of buttons, that used to flip the pages of data. If the book is not in stock at the moment, its row has a dark background in the table.



Code	Name
15274016	Life of Pi
15264732	Life of Pi
15123415	Royal Holiday

Fig. 5. Example of the client interface: book catalog

VIII. APPLICATION TESTING

The work of the client-server automation system of the library catalog was tested in three aspects: the correctness and efficiency of the server part (including load testing), the correctness of the system as a whole (integration testing), and the testing of individual components of the system.

The following classes were created as integration tests: a controller test responsible for requests to URLs related to viewing and making changes to the list of books; a login test; a registration test; a controller test responsible for requests to URLs related to the creation and modification of the parameters of pick-up requests and book return requests; controller test responsible for requests to URLs related to viewing the list of users and making changes to their profiles, both by the director and by the users themselves; test of the controller that redirects from the main page to the directory page.

The following classes were created as component tests: user service test; book service test; request service test.

Testing the client-server automation system of the library catalog allowed me to notice an error in the architecture of this system. Namely, when trying to write a test for the requests service, I realized that my way of passing objects from the client side of the system was wrong. Because only the object identifier was passed. Thus, the fields that stored other objects related to the received ones were empty. That made it impossible to test the corresponding component.

Load testing (see Table 1) checked the behavior of the system with a large number of requests. It showed the change in processed request per second (what also can be described as the speed of the system) and number of errors, that happens under load (increase of requests to the system). This part of testing is necessary to check the relative resistance of the developed system to overloads.

IX. COMPARATIVE ANALYSIS OF EXISTING LIBRARY AUTOMATION SYSTEMS

Koha is the first free software library automation package. Koha is distributed under the Free Software General Public License (GPL), version 3 or later.

Worldwide, almost all sizes of libraries and information centers use the Koha software for their day-to-day library work. It supports international catalog standards Full MARC21 and UNIMARC for professional cataloging, as well as major industry-standard database types RDBMS, SQL, and MySQL. It also supports printing barcodes [10]. Koha has the most features out of all library automation systems, and it’s configurable and adaptable. The problem is, to make any changes in the system software, you should know scripting and Perl. And installing the system requires good technical knowledge of Linux, MySQL & Apache [11]. Furthermore, the user interface for library workers is also pretty complicated, and because library workers must be trained to know how to use the software, the training costs becomes another disadvantage. It’s platform-independent on the client and server sides.

LibSys is most widely used in India, a fully integrated multi-user system design, to run on super, micro, and mini, computer under UNIX/VMS/LAN platforms. The main features of the software are the most advanced OPAC, which makes LibSys the most rewarding choice, for all librarians [10]. It is compatible with most international standards and supports a lot of languages. Libsys does not support the requirements of web links for electronic resources and facilitates to send the reports through e-mail [13]. It’s platform-independent on the client and server sides, but it’s also a commercial project, which means it has to be bought.

Table 1.

Results of server load testing

Number of requests	Number of unique requests	Processed requests per second	Number of errors	Number of errors, %
1	1	4	0	0
10	1	5	0	0
10	10	23	0	0
50	10	45	0	0
100	100	104	0	0
100	1	6	0	0
500	500	147	0	0
1000	500	155	0	0
3000	500	65	0	0
3000	3000	74	2	0.06
5000	5000	28	162	3.24
10000	5000	31	609	6
10000	10000	34	1825	18.25
20000	20000	35	3628	18.14
60000	30000	43	2830	4.71
60000	60000	42	9446	15.7

The NewGenLib, is an integrated library management system, developed by Verus Solutions Pvt Ltd. Kesavan Institute of Information and Knowledge Management in Hyderabad, is supporting and providing the domain and expertise [10]. It’s scalable, manageable, efficient, and uses chiefly open-source components. Together with allowing digital attachments to metadata. On the other side, it doesn't support some international metadata and interoperability standards don't support self-checkout, and has a lesser amount of statistic tools. [13]. It’s also platform-independent on the client and server sides.

SOUL (Software for University Libraries) is a state-of-the-art Integrated Library Management System (ILMS), designed and developed, based on the requirements of colleges and universities [10]. SOUL is known for its user-friendly interface. [13] It supports a lot of languages based on UNICODE and international catalog standards MARC21, AACR-2, MARCXML, NCIP, as well as different RDBMS, and cataloging of e-resources. On the other side, SOUL has some security issues [12]. SOUL is lacking article indexing and automatic tracking features [13]. It’s also a commercial project, and it’s not platform-independent.

Compared to Koha, SOUL, and LibSys, the proposed system is not commercial. It has a more user-friendly interface than Koha. And compared to SOUL, it’s platform-independent on the client and server sides and doesn’t have discovered security issues. It also supports self-checkout, unlike NewGenLib.

Overall, to compete with already established products, the system has to be secure and platform-independent on the client and server sides. Also, it has to be functional, and have a user-friendly interface.

X. CONCLUSION

The problem of developing a client-server system of library file automation was considered and its relevance was proved. The developed structure of the client-server automation system of the library index was given. The algorithm of its work was considered and its class diagram was given. Its architectural features were described. The structure of the database of the client-server system of automation of the library index was proposed. The peculiarities of the design of the user interface of the client-server automation system of the library index were considered. The question of testing the work of the developed system of automation of the library index was considered.

The developed client-server system for automating the library index allowed library employees to reduce the amount of effort required to audit books while helping the library cope with the constant growth of the amount of information related to the progress of mankind. In addition, automation reduced the number of errors in the work of library employees and allowed to ensure better data security and optimize their storage and search. In addition, the proposed system simplified the functions of monitoring the work of employees, and in general, facilitated all functions of managing a library institution. In addition, it also popularized the library institution and helped it in the functions of popularization and promotion of literature. This was achieved due to the user interface created within the project, in the form of a web page of the library, and providing users with access to its functions in online mode. At the same time, the client-server system of automation of the library card file simplified access to the functions of the library institution, due to a simple and intuitive user interface.

Thus, the developed client-server system for automating the library card file performed all the functions necessary for the work of the library and allowed it to automate the work, simplifying it, as well as popularizing the library institution itself.

In the course of testing, it was found that the developed client-server automation system of the library index was capable of ensuring error-free operation when receiving up to 3,000 simultaneous requests from unique users. However, even if more than 3,000 simultaneous unique requests were received, the probability of a server error did not increase by more than 18 %.

References

[1] M. Gupta and S. Jetty, (2018). "Library in Everyone's Pocket with reference to Bundelkhand University App", 5th ETTLIS conference, pp. 79–82. DOI: 10.1109/ETTLIS.2018.8485206

[2] Singh, S., & Malik, S. (2018). Use of Mobile Technology in University Libraries A Case Study of Maharaja Jiwaji Rao Library, Vikram University, Ujjain (M.P.). 2018 5th International Symposium on Emerging Trends and Technologies in Libraries and Information Services (ETTLIS), pp. 235–239 DOI: 10.1109/ettlis.2018.8485260

[3] Kim, J. H., Hee Lee, J., & Lee, K. J. (2021). A Study on the Issues Related to Building a Library Information System Based on Deep Learning. 2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter), pp. 287–289 DOI: 10.1109/SNPDWinter52325.2021.00076

[4] Puritat, K., & Intawong, K. (2020). Development of an Open Source Automated Library System with Book Recommendation System for Small Libraries. 2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT &amp; NCON), pp. 128–132 DOI: 10.1109/ectidamtncon48261.2020.9090753

[5] W3Techs. (2022). Usage statistics of operating systems for websites. [online] Available: https://w3techs.com/technologies/overview/operating_system (Accessed: 25 September 2022).

[6] Lang, A. W. (2016). John Palfrey, BiblioTech: Why Libraries Matter More than Ever in the Age of Google.

[7] Kundys, S. Havano, B. Morozov, M. (2022). “Software System for Monitoring the Situation in the Settlement”, Advances in Cyber-Physical Systems, 7(1), pp. 38–45. DOI: 10.23939/acps2022.01.038

[8] Muniraja, A. (2021). Library Automation–An Introduction. International Journal of Research in Library Science, 7(2), pp. 192–198. DOI: 10.26761/IJRLS.7.2.2021.1413

[9] Wiseso, L. G., Imrona, M., & Alamsyah, A. (2020). Performance Analysis of Neo4j, MongoDB, and PostgreSQL on 2019 National Election Big Data Management Database. 2020 6th International Conference on Science in Information Technology (ICSITech), pp. 91–96. DOI: 10.1109/ICSITech49800.2020.9392041

[10] Naik, U. (2016). Library Automation Software: A Comparative Study of Koha, Lib Sys, New Gen Lib and SOUL. International Journal of Library Science and Research IJLSR, 6(6), pp. 77–86. DOI: 10.24247/ijlsrdec20168

[11] LIBCON. (2022). KOHA for Library automation, benefits & limitations. [online] Available: https://libcon.in/blog/KOHA-for-automation.aspx (Accessed: 24 September 2022).

[12] Najan, S., Gawde, M., Solnki, M. S., & Mishra, D. K. (2012). Comparative study of library automation software (Special reference of SOUL and TLSS). International journal of library automation, networking and consortia, 1(1), pp. 1–15.

[13] Kumar, S., & Prasad, H. N. (2021). Evaluation of Integrated Library Automation Software in Libraries: A Comparative Study

of SOUL, NewGenLib and Libsys. Library Waves, 7(2), pp. 59–70. ISSN: 2455-2291

Nazar Repianskyi is a student of Lviv Polytechnic National University. In 2021 he received a Bachelor's degree in Computer Engineering at Lviv Polytechnic National University. Now he is getting a Master’s Degree in the field of Computer Systems and Networks.

His research interests include web technologies, data processing, and back-end development using Spring Framework.

Taras Rak is a professor at Lviv Polytechnic National University, Vice-rector, and Professor at IT STEP University.

A graduate of Lviv Polytechnic State University, 1996, specialty “Computer and Intelligent Systems and Networks”, honors degree. Candidate of Technical Sciences, 2005, specialty “Systems Analysis and Theory of Optimal Solutions”. Since 2014 – Doctor of Technical Sciences, specialty “Information Technologies”.

Taras Rak is an author of over 150 scientific and educational publications.