MSC 68T05, 68M15, 68M20 **DOI: 10.14529/mmp210304**

INTELLECTUAL MATHEMATICAL SUPPORT SOFTWARE AND INNER ARCHITECTURE OF LMS MAI CLASS.NET

***E.A. Zharkov*[1], *V.D. Malygin*[2]**
[1]Freight One, Moscow, Russian Federation
[2]MOC IKT, Krasnogorsk, Russian Federation
E-mail: zharkovevgeniy94@gmail.com, vladislavmalygin@gmail.com

Distance education prove to be effective in improving the learning and teaching environment. One of the main advantages of distance learning is that web-based courses can be taken anytime and anywhere. The implementation of an e-learning management system (LMS) requires not only good and fast hardware, but also the use of modern software technologies and architectural solutions. This article outlines the main ways of forming the LMS architecture based on a microservice approach, which allows the achievement of high performance and fault tolerance. A distinctive feature of the CLASS.NET system is the presence of a special mathematical software package that allows the optimization of educational processes and tasks (such as students tests generation, students progress analysis, knowledge level assessment, task difficulty analysis, personal learning curve planning). The process of interaction between the LMS system and mathematical software package, as well as the main ways of forming such software as completely independent applications for their further integration into other learning management systems, are thoroughly described. The efficiency of the microservice architecture in terms of scaling, performance and general behavior in case of critical errors in comparison with other systems based on classical architectural approaches is shown. The algorithm of predicting the time a student spends to answer the tasks, which is included in the mathematical software package, is considered.

Keywords: learning management system; LMS, e-learning; adaptive learning curve; microservices.

Dedicated to Professor A.I. Kibzun
on the occasion of his anniversary

Introduction

Distance learning system (Learning Management System, LMS) is a separate class of information systems. In addition to its application as a system for managing and publishing information in various forms (theoretical material, presentations, video files etc.), LMS makes it possible to organize and track educational classes, workload and student progress. Also, LMS implies some level of integration with other information systems, for example, with the information base of the dean office to transmit data on the successful completion of the course or the current progress of the student. A distinctive feature of the Moscow Aviation Institute LMS CLASS.NET [1,2] is the presence of a special mathematical application that allows to use the methods of mathematical statistics to solve a number of problems such as: students knowledge level assessment based on his or her answers to sets of test problems, formation of levels of difficulty of problems [3], generation of sets of test problems with a specific level of difficulty and time limit [4], predicting the time of students response to a specific problem, as well as collecting user activity data. Thus, LMS is one of the components of a complex distributed system and solves the problems of related classes of information systems such as a content management system, a data analysis collection system, a system of expert decisions.

It is obvious that the main difficulty in building such an information system is to organize the effective interaction of all its internal components. Since the goals of the LMS are narrowly focused, most of the effective solutions presented in the modern IT market have a closed architecture and solve the problems of a particular enterprise or business. And the use of open distance learning systems is impractical due to the low performance quality and lack of proper functionality.

It is also worth mentioning that an important factor in the modern world of information technology is that the quality of open methods of organizing application architectures is increased, as well as design standards and new protocols for data transfer and synchronization are appeared.

Therefore, expanding or improving the available open source LMS products takes a significant amount of effort comparable to developing own system.

These factors form the task of developing an effective LMS adapted to the tasks of content formation, data analysis and integration, relevant.

1. Architecture of LMS MAI CLASS.NET Application

Before proceeding to consideration of the possible methods to develop any application, it is necessary to determine the general tasks that the LMS should solve. The main tasks that should be highlighted are as follows:

- ability to manage the life cycle of posted information (tests, lecture texts, end of unit tests, etc.): Publish or remove the publication by a schedule; Automatic archiving of obsolete information; Possibility of limited access (for example, information is available only after passing the test);
- ability to integrate content from external data sources: Support of links to external sources of information; Using embedded content (iframe, embed); Interaction with the databases of other systems;
- combination of the possibility to post both static and interactive information: Ability to use special means of information display control (wysiwyg - editor, placement of video materials, presentation files, interactive data graphs); Tests and tasks builder (using various methods of answer input, time constraints, etc.);
- user authorization: Ability to use different roles to restrict access (e.g. student, teacher, moderator, admin); Possibility of end-to-end authorization through certain external systems, or registration with data confirmation;
- presence of modern standards of application programming interfaces (API) for interaction with external systems.

Based on the list of the given tasks, we can confidently conclude that this system is quite complex and necessarily requires a well-developed architecture, not only for the possibility of its expansion, but also for maintaining operability.

The work [1] describes the structure of the LMS CLASS.NET, which is shown in Fig. 1.

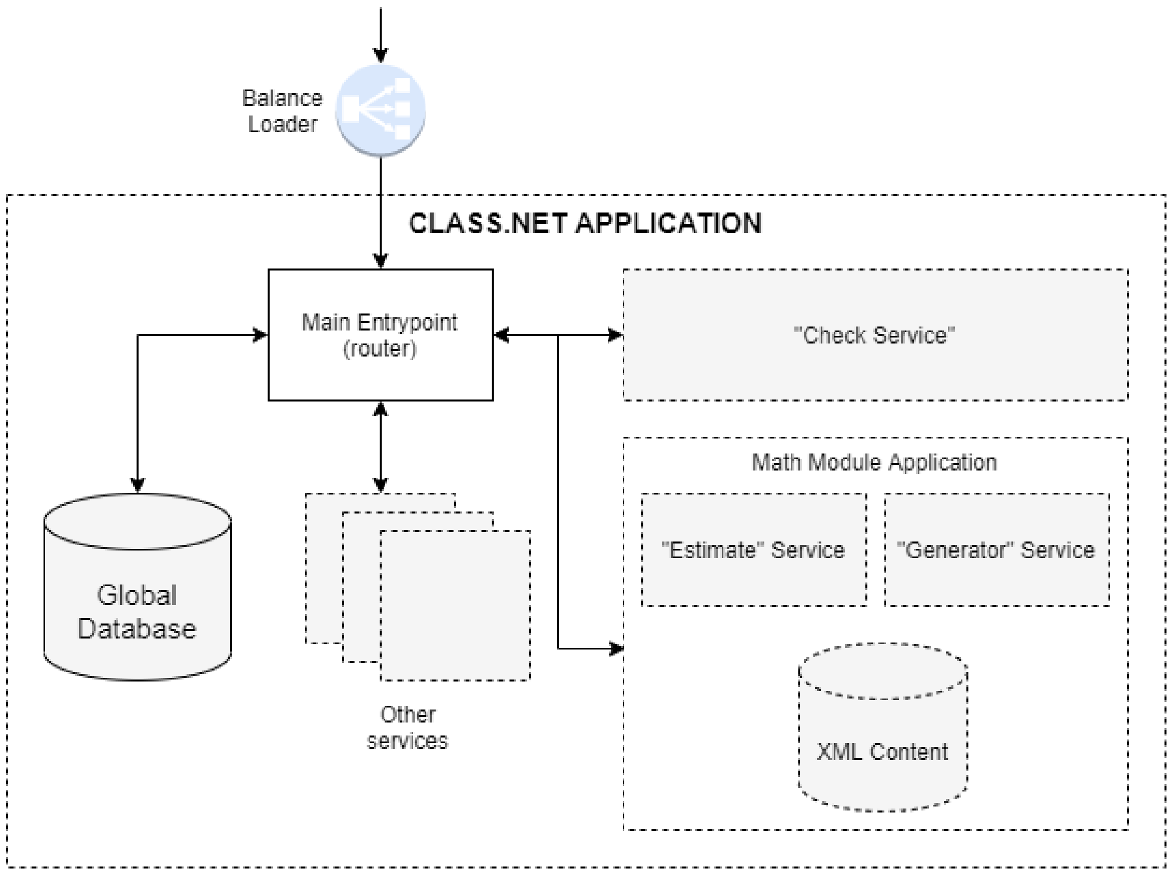

Fig. 1. Monolithic architecture of the LMS MAI CLASS.NET

The software part has a client-server architecture, where the requested course is displayed on the client side, and information is processed and stored in the corresponding database on the server side. The main idea of this architecture is based on the so-called “monolithic” approach [5, 6], when the entire system is deployed as one single application. At the same time, there are no restrictions in the choice of development tools. The CLASS.NET application consists of a hierarchy of directories (services), each of which is responsible for a corresponding service (user management, generator, evaluator, task complexity estimation module).

Such application is run by gunicorn in a docker container. HTTP requests to the web application are processed by the Apache server in conjunction with Nginx. With gunicorn flexible control configurations, we can run multiple instances of an application in multi-threaded mode along with a balance loader to increase scalability and improve quality of availability.

This architectural solution has a number of undeniable advantages [5–7]:

- Simplicity of development, i.e. the entire application is available within a single code base;
- Simplicity of deployment that means it is necessary to deploy the executable file of the process in the appropriate run-time environment;

- Easy to scale, i.e. run multiple instances of the application together with the balance loader.

However, as an application grows large due to increased functional requirements for the system and an increase in the size of the development team, a number of deficiencies that significantly affect the overall health of the system becomes noticeable [5–7].

- The volume of monolithic codebase increases significantly. As a result, the degree of further development and expansion of the system usually decreases. In addition, since there are no hard boundaries between individual logical services, the modularity of the system is broken over time, which again leads to the difficulty of implementation of new changes and to an overall degradation of the code quality. As a result, system responsiveness suffers.
- Monolith approach implies a long-term commitment to a specific set of technologies and tools chosen during the initial development phase. And in case of platform obsolescence, there may be a problem of a gradual migration of the application to a newer and better environment. It is possible that in order to implement a newer platform, the entire application will have to be rewritten, which certainly entails certain risks.

The LMS MAI CLASS.NET has a special module for mathematical support. This module is a set of statistical algorithms that allow solving optimization and machine learning problems to obtain certain data that allows to optimize the educational process and form an individual learning path. The main features of this module are as follows.

- Generate the unique variants of tests using random parameters.
- Analyse the symbolic way of answering. When the user enters a mathematical formula as an answer, the module analyzes the correctness of the answer by comparing the function value by points with the reference one.
- Generate the individual sets of tests with a certain level of difficulty and time limit.
- Evaluate the students knowledge level and the level of difficulty of the tasks in relation to all students answers.

The main problem of this module is that when a certain algorithm was launched, this module overtook most system resources, thereby reducing overall performance. Among other things, to collect statistics for any user activity, part of the data was constantly written to the database, respectively, multiple single transactions also slowed down the speed of the database. In order to solve the arising problems of scaling the system, it was decided to use the microservice architectural approach [5, 6] to split the monolith application into certain parts, i.e. the so-called services.

2. Microservice Architecture of the LMS MAI CLASS.NET Application

It is necessary to define an architecture that represents the system as a set of loosely coupled interacting services. Each service should:

- be easy to maintain and test; ensure fast and frequent deployment;

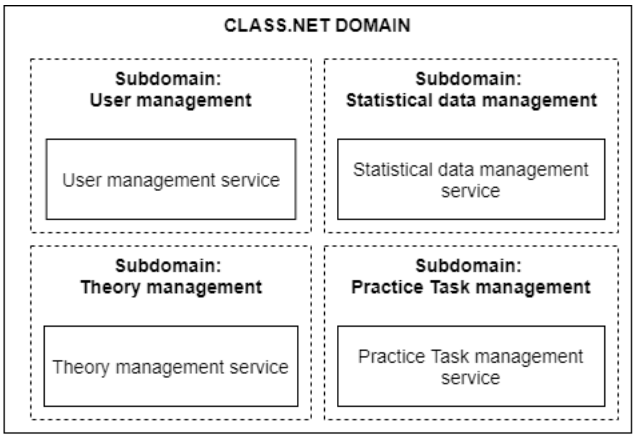
CLASS.NET DOMAIN
Subdomain: User management
User management service
Subdomain: Statistical data management
Statistical data management service
Subdomain: Theory management
Theory management service
Subdomain: Practice Task management
Practice Task management service

Fig. 2. Domain Driven Design of the LMS MAI CLASS.NET

- have a loose coupling with other services, which allows developers to work on their modules most of the time independently, without being affected by changes in other services;
- have the ability to self-deploy, therefore, each service can independently connect to the general process of the system;
- use either a synchronous protocol such as HTTP / REST or an asynchronous protocol such as AMPQ;
- have its own database that is one of the key, but not mandatory, requirements for the service, and this requires strict data consistency between services.

The microservice architecture approach is often compared to the service-oriented approach. The difference in these architectural solutions is described in detail in [8]. For LMS MAI CLASS.NET, it is proposed to use a mixed approach of two solutions, which allows to use the most suitable solution in the corresponding parts of the system.

In order to define each service, it is suggested to use Domain Driven Design (DDD) [9]. The idea behind this approach is that services are separated based on functional processes in the application domain. Thus, there is a main area of operation of a specific part of the system where "small" application is placed and solves problems that should not go beyond the boundaries of this area. The subject area of an application is called a domain, and all its subareas of operation are called subdomains. Thus, each service strictly fulfills its task and obeys a certain logic of the subject area, which does not go beyond the subdomain. The division of the CLASS.NET LMS into domains by subject area is shown in Fig. 2).

Each subdomain has its own Bounded Context [9]. This bounded context is isolated from other parts of the system and the interaction of different subdomains is carried out using the API in conjunction with the Event Bus.

As it is shown in representation of the CLASS.NET domain, the "Statistical Support" subdomain is a completely isolated layer, which allows to transfer the mathematical support module to a separate web service. This architectural solution is due to the service-oriented approach, since the main functionality of the LMS does not depend on the running of this web service.

As a gradual migration of a monolithic system to a new architecture, it is recommended to use the Anti-Corruption Layer [10]. This is a special layer, which is used as an interface

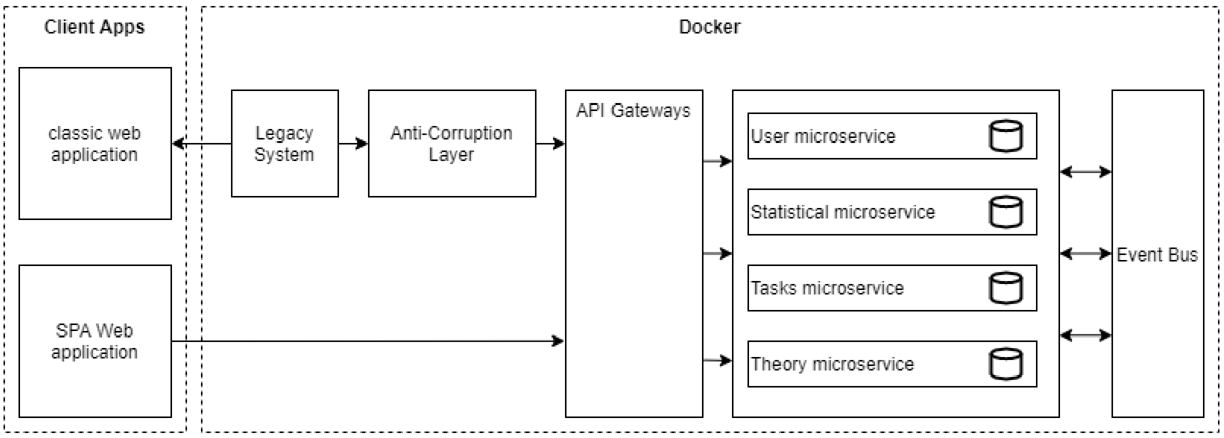

Fig. 3. Microservice architecture of the LMS MAI CLASS.NET

for interaction between the Legacy system and new microservices. This approach allows to gradually update the necessary functionality of the system and redirect one part of incoming requests to a specific microservice, and process the other part of requests according to the classic scenario. The main interaction between the client and the system is implemented using the API Gateway template [10], which includes a set of special smart endpoints that redirect requests to the appropriate microservices.

Therefore, the monolithic application was gradually divided into corresponding services and the final architecture of the CLASS.NET LMS is shown in Fig. 3.

A microservice architectural solution certainly has a positive effect on ensuring continuous development and deployment of large and complex applications and has the following advantages:

- high system availability, i.e. each service or application in the system is relatively small and therefore easier to understand and change;
- better test-ability, i.e. the testing process is reduced to separate independent unit testing of each service;
- services can be deployed independently of each other, allowing development by multiple autonomous development teams, where each team owns and is responsible for one or more services, and can develop, test, deploy, and scale their modules independently of other teams.

But despite of its effectiveness, this architectural solution has the following drawbacks.

- When developing, it is necessary to take into account the features of the distributed system.
- It is necessary to implement a specific logic for communication and synchronization between services.
- Testing the system is conducted under conditions of partial failure.
- Implementing requests that cover multiple services require careful coordination between development teams. Each service has to be available at the time of the request.
- There is increased consumption of physical resources. Each service is a separate application, which means it must be located on either a separate server or a virtual machine. This significantly increases the equipment costs.

- Due to the use of local databases, data consistency must be maintained at all times. Maintaining data consistency between services can be extremely challenging due to constant changes and critical crashes.

3. Results of Stress Tests

In order to evaluate the performance of the system based on the two different architectural solutions, several load testing scenarios were formed:

- Test A: Gradually increasing the number of users from 1 to 100 within 10 minutes and then holding the load for 5 minutes;
- Test B: Gradually increasing the number of users from 100 to 1000 within 10 minutes and then holding the load for 5 minutes;
- Test C: Constant load of the main system with 100 users and a parallel call to the math support service.

Here each user generated from 1 to 5 server requests. Tests were carried out on applications running in docker containers with 16 GB of memory and 3.0 GHz of CPU. JMeter [11] application was used for testing, where the average response time parameter was used as the main metric (the time between sending a request by the client and receiving a response from the server) [12].

According to the application statistics for the CLASS.NET LMS, the maximum number of requests per second does not exceed 600. Therefore, "Stress Test A" is intended to check the system performance in normal mode. The results of the stress test are presented in Fig. 4.

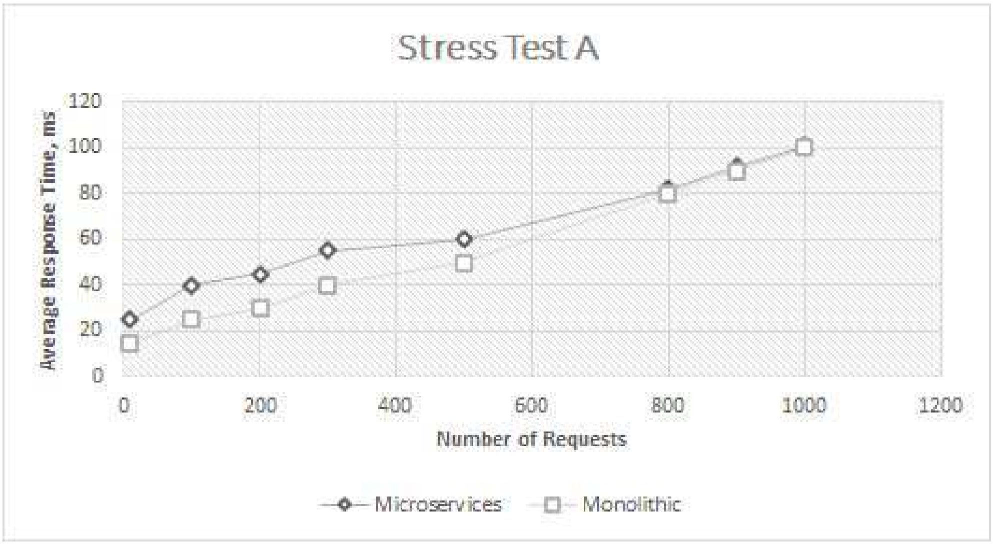

Fig. 4. Stress Test A

Under the condition of the small amount user activity, the monolithic architecture processed requests faster than a microservice architecture. The difference in performance, on average, is 14%, but taking into account the fact that the maximum response time does not exceed 100 ms, the response is considered acceptable, and the performance difference is not significant.

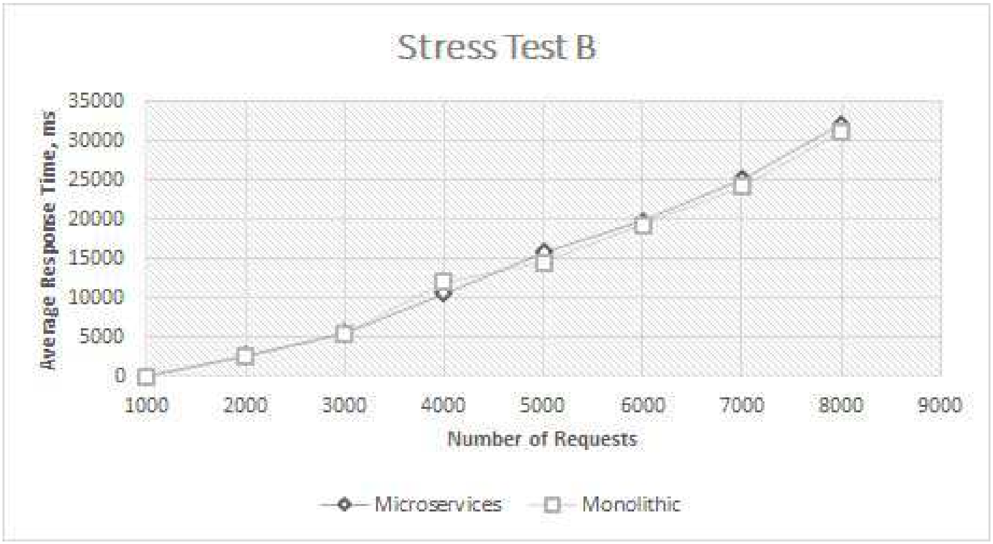

Fig. 5. Stress Test B

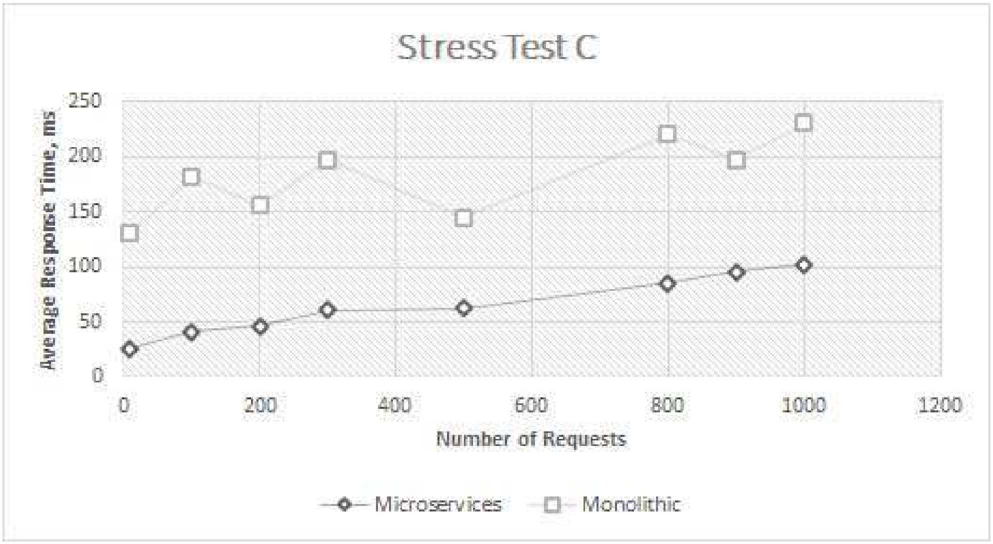

Fig. 6. Stress Test C

The results of "Stress test B" are presented in Fig. 5. This stress test practically emulates a DDoS attack, which leads to an average response time of 30 seconds. However, the average performance of architectures remains approximately equal, where the difference does not exceed 4%.

Stress test C repeats Stress test A scenario, but the condition for the active use of the math support module is added. This test is designed to check the system performance in normal mode while the algorithms of the mathematical package are running. The test results are shown in Fig. 6.

As it is shown, the average response time of a monolithic system is increased significantly, while the response time of a microservice system is practically not changed. These results are quite expected, since the mathematical support module is extremely resource-demanding, and if the server does not physically have the necessary resources, the request processing time will only increase.

Based on the results obtained, we can make an unambiguous conclusion that the monolithic architecture is better suited for small applications and has better performance under the low user activity conditions. However, under high load conditions, both systems show almost the same performance. The results of this stress testing coincide with the results obtained in [13]. The stress test C showed an unambiguous increase in performance under the condition of active use of the mathematical support module, but this only confirms the fact that, by allocating separate computing resources to a demanding module, it will not affect the main system.

4. Predicting Time a Student Spends to Answer Tasks During Testing

When users interact with assessment questions in an online course, the data that usually gets the most attention is the answers sent, and sometimes only the correctness of these answers or their score received. However, there may be unknown external factors that distort the students level of knowledge under remote test conditions, so the time a user spends on the question is also important: it is perhaps the most easily obtained data that reveals something about the process by which the user came to the answer.

The analysis of the time of response to the question allows to quantify some properties of the questions (how long a question usually takes and how complex it is), as well as some user properties (who exactly solved this problem and whether the answers to the task were known in advance). The question properties are relevant to course design, and the user experience speed can be related to the user ability and preferred mode of interaction with the course. Extracting such parameters requires a parametric statistical model of the response time [18].

To solve this problem, an additional service was added that collects analytical data of user activity. Main attributes to collect are as follows:
1) total number of sessions;
2) amount of time spent in the learning environment;
3) total number of visits to training materials;
4) total number of visits to the hint pages;
5) total number of visits to the pages of tests;
6) average amount of time (in seconds) spent writing the test;
7) average number of keystrokes made by a student while writing the test;
8) average number of backspace keys used by the student when writing the test;
9) average number of times the page lost focus while writing the test;
10) average number of changes in the choice of different answers made by the student during writing of the test;
11) average complexity of the tests;
12) average number of errors;
13) average time spent on pages with training materials;
14) total number of hints to open;
15) average student performance score during the course of study;
16) average number of missed tests.

Problem Statement. Determine the value of the output variable Y that is the time taken to pass the test from the input independent variable X:

$$Y = BX + U,$$

where U is the error matrix, and B is the coefficients that are calculated as a result of performing the regression analysis. This model has both positive and negative sides. Positive aspects are as follows:

- model is very fast, can run on very large datasets;
- coefficients before the features can be interpreted, provided that the features are scaled;
- model is simple with a well-interpreted result of the influence of each attribute on the model operation.

Negative aspects are as follows:

- model does not work well in problems where the dependence of responses on features is complex and nonlinear;
- with a small number of features, the model does not take into account all the dependencies to output accurate results;
- in practice, the assumptions of the Markov–Gauss theorem are almost never fulfilled, so more often linear methods perform worse than, for example, SVM and ensembles.

Several machine learning algorithms were used to predict continuous variables: linear regression, decision tree, random forest, and ensemble of decision trees (xgboost) [14]. Table shows the results of applying these algorithms for the estimated metrics of root-mean-square error (RMSE) and coefficient of determination R^2. According to the results of the algorithms, the ensemble of decision trees gives the highest accuracy in comparison with other machine learning algorithms.

Table

The algorithms comparison

Linear Regression	$0,74$	$38,87$
Random Tree	$0,79$	$33,03$
Random Forest	$0,79$	$29,34$
XGBoost	$0,82$	$28,35$

To evaluate the accuracy of the machine learning model, take the mathematical hierarchical model of Van der Linden [15]. In this method, the measurement of students abilities is based on a three-parameter logistic model with random element parameters for accuracy and for response time. The model is a hierarchical structure due to the multidimensional distribution for abilities and speed, as well as for element parameters such as accuracy and response time. The covariance matrices describe the relationships between these parameters, and the mean vectors describe the differences between people and objects [17]. This model uses a log-normal distribution, which is well studied, quite simple to work with, and has qualitatively correct characteristics: no negative values in the distribution area, one peak, and a long tail on the right.

Also, the central limit theorem states that under certain mild conditions, the sum of a very large number of independent random variables approaches the normal distribution. Accordingly, the response time model is defined as follows [16]:

$$log(T_{ij}) = \beta_i - \tau_j + \varepsilon_{ij},\ \varepsilon_{ij} \sim \mathbb{N}(0, \alpha_i^{-2}), \tag{1}$$

where β_i is the time intensity parameter for the i-th task, τ_j is the speed parameter for the j-th person, and the error ε_{ij} is assumed to be normally distributed with an average value 0 and inverse variance, α_i^{-2}. In addition, α_i denotes the time discrimination parameter of a particular element that quantifies the variability of the $log(T_{ij})$ distribution. The histogram for the time that students spent to answer the test, and the distributions obtained using a mathematical model and a machine learning model are shown in Fig 7. According to the user response time histogram, the data is distributed normally with numerous outliers. At the beginning of the histogram, it is observed that a small part of the students either viewed the test and immediately closed it, or quickly put down random answers to questions. The main group of students falls into the peak of the normal distribution. Also, a small group of students spends all the available test time to pass it.

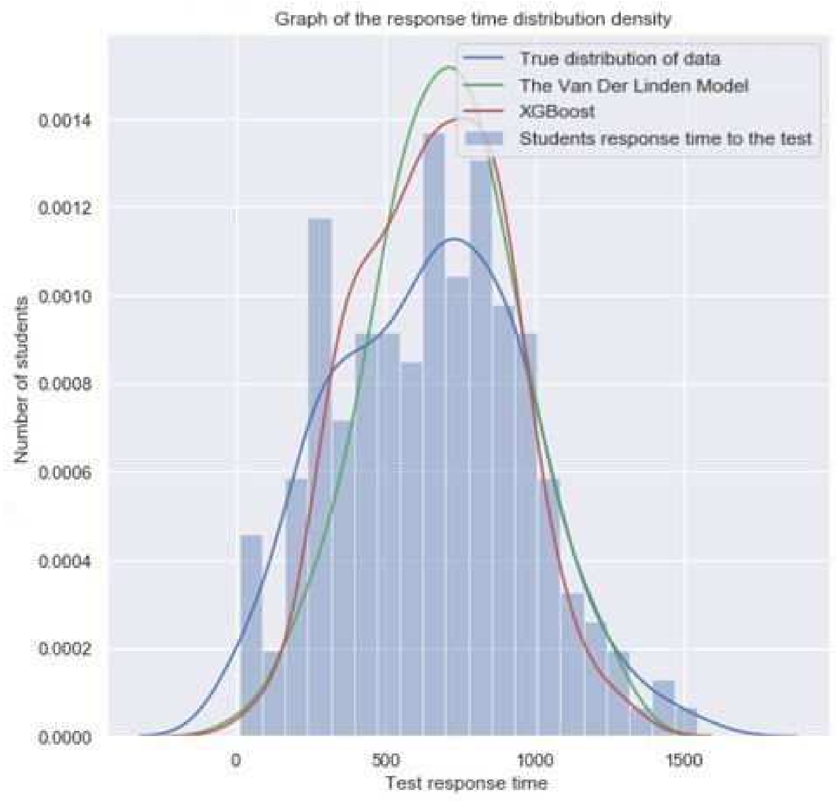

Fig. 7. Response time distribution density

In order to test the assumption of normality of the data, we use the Shapiro–Wilk test. The null hypothesis H_0 of this test is that the sample came from a normally distributed population. An alternative hypothesis of H_1 is that the distribution law is not normal. Using the built-in stats.shapiro function from the scipy mathematical library, the test statistics value $W = 0,989$ and the approximation for the resulting statistics value $p - value = 0,126$ were obtained. Accordingly, at the significance level $\alpha = 0,05$, the hypothesis H_0 should be accepted (since $p - value > \alpha$). According to the results obtained, the ensemble of decision trees predicts the students time of response to the problem no worse than the values obtained on the basis of the Vander model, which indirectly allows to conclude that the trained model is correct. Thus, to identify students with ready answers to the test, it is necessary to predict the students response time, find the left boundary of the one-way interval at the confidence level $p = 0,05$ (the quantile level p), and if the response time is less than the obtained left boundary, then we consider the task compromised.

The described response time model can be used to evaluate the characteristics of both the trainees (slowness) and the tasks themselves (complexity). It was also found that, in self-directed online courses from our dataset, higher user sluggishness is associated with higher achievement levels: students who take their time tend to do better. Because slowness (or some other metric that reflects the students response time) can be tracked by the course teacher in the course analytics in addition to other performance metrics for tagging purposes. Our results show that faster and lower-performing learners cause more anxiety than slower learners.

Conclusion

The article reviews the main aspects of creating an LMS and identifies the main problems associated with the architecture of such complex information systems. Also, we describe the process of migrating to the microservice architecture and reveal the main advantages and disadvantages of this solution. Despite the small differences in performance, the use of LMS MAI CLASS.NET in production confirms the effectiveness and reliability of the microservice architecture approach.

Thanks to the task response time estimation algorithm, it becomes possible to analyze user activity and identify students who had prepared the correct answers in advance, as well as to find tasks that were previously compromised, and remove them from the set of tasks or replace them with similar tasks.

References

1. Naumov A.V., Dzhumurat A.S., Inozemtsev A.O. [Distance Learning System for Mathematical Disciplines CLASS.NET]. *Herald of Computer and Information Technologies*, 2014, vol. 1, no. 10, pp. 36–40. (in Russian)

2. Kibzun A.I., Karolinskaya S.N., Shayukov R.I. [A Remote Study System for Mathematical Disciplines in the University]. *Herald of Computer and Information Technologies*, 2006, vol. 1, no. 4, pp. 29–36. (in Russian)

3. Kibzun A.I., Zharkov E.A. Two Algorithms for Estimating Test Complexity Levels. *Automation and Remote Control*, 2017, vol. 78, no. 12, pp. 2165–2177.

4. Naumov A.V., Mkhitaryan G.A. On the Problem of Probabilistic Optimization for Tests within the Time-Limit. *Automation and Remote Control*, 2016, vol. 77, no. 9, pp. 1612–1621. DOI: 10.1134/S0005117916090083

5. Rui Chen, Shanshan Li, Zheng Li. From Monolith to Microservices: A Dataflow-Driven Approach. *24th Asia-Pacific Software Engineering Conference (APSEC)*, Nanjing, 2017, pp. 466–475.
6. Newman S. *Building Microservices: Designing Fine-Grained Systems*, Sebastopol, O'Reilly Media, 2015.
7. Villamizar M., Garces O.,Lang M. Cost Comparison of Running Web Applications in the Cloud Using Monolithic, Microservice, and AWS Lambda Architectures. *Service Oriented Computing and Applications*, 2017, vol. 11, no. 2, pp. 233–247.
8. Richards M. *Microservices vs. Service-Oriented Architecture*, Sebastopol, O'Reilly Media, 2015.
9. Vernon V. *Implementing Domain-Driven Design*, Massachusetts, Addison-Wesley Professional, 2013.
10. Richardson C. *Microservice Patterns*, New York, Manning, 2017.
11. *Apache JMeterTM (2021).* Available at: https://jmeter.apache.org/ (accessed 18 April 2021).
12. Kumari S., Rath S. Performance Comparison of SOAP and REST Based Web Services for Enterprise Application Integration. *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Bangalore, 2015, pp. 1656–1660.
13. Ueda T., Nakaike T., Ohara M. Workload Characterization for Microservices. *IEEE International Symposium on Workload Characterization (IISWC)*, Rhode Island, 2016, pp. 1–10.
14. *XGBoost Homepage (2021).* Available at: https://neerc.ifmo.ru/wiki/index.php?title=XGBoost (accessed 18 April 2021).
15. Van der Linden W.J. *Conceptual Issues in Response-Time Modeling*, Philadelphia, Law School Admission Council, 2008.
16. Van der Linden W.J. A Bivariate Lognormal Response-Time Model for the Detection of Collusion Between Test Takers. *Journal of Educational and Behavioral Statistics*, 2009, vol. 34, no. 3, pp. 378–394.
17. Liao D. *Modeling the Speed-Accuracy-Difficulty Interaction in Joint Modeling of Responses and Response Time: Dissertation*, Maryland, University of Maryland, 2018. DOI: 10.13016/M25H7BX5P
18. Rushkin I., Chuang I., Tingley D. Modelling and Using Response Times in Online Courses. *Journal of Learning Analytics*, 2019, vol. 6, no. 3, pp. 76–89.

Received April 27, 2021

УДК 004.416.6 **DOI: 10.14529/mmp210304**

ВНУТРЕННЯЯ АРХИТЕКТУРА СДО МАИ CLASS.NET И ЕЕ КОМПЛЕКС МАТЕМАТИЧЕСКОЙ ПОДДЕРЖКИ

Е.А. Жарков[1]***, В.Д. Малыгин***[2]
[1]ПАО «Первая Грузовая Компания», г. Москва, Российская Федерация
[2]ГКУ МО «МОЦ ИКТ», г. Красногорск, Российская Федерация

Дистанционное образование доказало свою эффективность в улучшении условий обучения и преподавания. Одним из главных преимуществ дистанционного обучения является то, что веб-курсы можно получить в любое удобное время и в любом удобном месте. Для внедрения системы электронного обучения требуется не только хорошее и быстрое оборудование, но и использование современных программных технологий и архитектурных решений. В данной статье изложены основные способы формирования архитектуры СДО на основе микросервисного подхода, который позволяет добиться высокой производительности и отказоустойчивости. Отличительной особенностью системы CLASS.NET является наличие специального математического программного комплекса, позволяющего оптимизировать учебные процессы и задачи (формирование контрольных работ для студентов, анализ успеваемости учащихся и уровень их знаний, анализ трудности задач, формирование персональной траектории обучения). Подробно описан процесс взаимодействия основной системы LMS с данным математическим программным комплексом, а также основные способы формирования таких программных комплексов как полностью независимых приложений для их дальнейшей интеграции в других системы обучения. Показана эффективность микросервисной архитектуры с точки зрения масштабирования, производительности и общего поведения в случае критических ошибок, при сравнении ее с другими системами, основанными на классических архитектурных подходах. Рассматривается модель прогнозирования ответа на тестовое задание, входящее в математический программный комплекс.

Ключевые слова: система дистанционного обучения; монолитная архитектура; микросервисная архитектура; адаптивная кривая обучения.

Литература

1. Наумов, А.В. Система дистанционного обучения математическим дисциплинам CLASS.NET / А.В. Наумов, А.С. Джумурат, А.О. Иноземцев // Вестник компьютерных и информационных технологий. – 2014. – Т. 1, № 10. – С. 36–40.
2. Кибзун, А.И. Система дистанционного обучения по математическим дисциплинам в вузе / А.И. Кибзун, С.Н. Каролинская, Р.И. Шаюков // Вестник компьютерных и информационных технологий. – 2006. – Т. 1, № 4. – С. 29–36.
3. Кибзун, А.И. Два алгоритма оценивания уровней сложности тестов / А.И. Кибзун, Е.А. Жарков // Автоматика и телемеханика. – 2017. – Т. 1, № 12. – С. 84–99.
4. Наумов, А.В. О задаче вероятностной оптимизации для ограниченного по времени тестирования / А.В. Наумов, Г.А. Мхитарян // Автоматика и телемеханика. – 2016. – Т. 6, № 9. – С. 124–135.
5. Rui Chen. From Monolith to Microservices: A Dataflow-Driven Approach / Rui Chen, Shanshan Li, Zheng Li // 24th Asia-Pacific Software Engineering Conference (APSEC). – 2017. – P. 466–475.
6. Newman, S. Building Microservices: Designing Fine-Grained Systems / S. Newman. – Sebastopol: O'Reilly Media, 2015.
7. Villamizar, M. Cost Comparison of Running Web Applications in the Cloud Using Monolithic, Microservice, and Aws Lambda Architectures / M. Villamizar // Service Oriented Computing and Applications. – 2017. – V. 11, № 2. – P. 233–247.
8. Richards, M. Microservices vs. Service-Oriented Architecture / M. Richards. – Sebastopol: O'Reilly Media, 2015.
9. Vernon, V. Implementing Domain-Driven Design / V. Vernon – Massachusetts: Addison-Wesley Professional, 2013.
10. Richardson, C. Microservice Patterns / C. Richardson – New York: Manning Publications, 2017.
11. Apache JMeterTM. – URL: https://jmeter.apache.org (дата обращения 18.04.2021).

12. Kumari, S. Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration / S. Kumari, S.K. Rath // International Conference on Advances in Computing, Communications and Informatics (ICACCI). – Bangalore, 2015. – P. 1656–1660.
13. Ueda, T. Workload characterization for microservices / T. Ueda, T. Nakaike, M. Ohara // IEEE International Symposium on Workload Characterization (IISWC). – Rhode Island, 2016. – P. 1–10.
14. XGBoost Homepage. – URL: https://neerc.ifmo.ru/wiki/index.php?title=XGBoost (дата обращения 18.04.2021).
15. Linden, W. Conceptual Issues in Response Time Modeling / W. Linden // Journal of Educational Measurement. – 2009. – T. 46, № 3. – P.247–272.
16. Linden, W. A Bivariate Lognormal Response-Time Model for the Detection of Collusion Between Test Takers / W. Linden // Journal of Educational and Behavioral Statistics. – 2009. – V. 34, № 3. – P. 378–394.
17. Liao, D. Modeling the Speed-Accuracy-Difficulty Interaction in Joint Modeling of Responses and Response Time: Dissertation / D. Liao. – Maryland: University of Maryland, 2018.
18. Rushkin, I. Modelling and Using Response Times in Online Courses / I. Rushkin, I. Chuang, D. Tingley // Journal of Learning Analytics. – 2019. – V. 6, № 3. – P. 76–89.

Евгений Александрович Жарков, ПАО «Первая Грузовая Компания» (г. Москва, Российская Федерация), zharkovevgeniy94@gmail.com.

Владислав Дмитриевич Малыгин, ГКУ МО «МОЦ ИКТ» (г. Красногорск, Российская Федерация), vladislavmalygin@gmail.com.

Поступила в редакцию 27 апреля 2021 г.