UDC 004.75:378.14

A.O. Lavreniuk
V.P. Maidaniuk

ARCHITECTURAL APPROACHES TO BUILDING CLIENT-SERVER SYSTEMS FOR EDUCATIONAL PLATFORMS

Vinnytsia National Technical University

Анотація

У тезах розглядаються основні архітектурні підходи до будівництва клієнт-серверних систем для освітніх платформ. Аналізуються монолітна, мікросервісна та сервісно-орієнтована архітектури, визначаються їхні переваги та недоліки в контексті електронного навчання. Особлива увага приділяється вимогам до масштабованості, надійності та безпеки систем, що орієнтовані на розповсюдження та обмін освітніми матеріалами.

Ключові слова: клієнт-серверна архітектура, освітня платформа, мікросервіси, REST API, хмарні обчислення, масштабованість.

Abstract

The article examines the main architectural approaches to building client-server systems for educational platforms. Monolithic, microservice, and service-oriented architectures are analyzed, their advantages and disadvantages in the context of e-learning are identified. Special attention is paid to scalability, reliability, and security requirements for systems oriented towards the distribution and exchange of educational materials.

Keywords: client-server architecture, educational platform, microservices, REST API, cloud computing, scalability.

Introduction

The rapid expansion of digital technologies in education has driven significant demand for robust, scalable platforms capable of managing and distributing educational content effectively. Client-server architecture has become the foundational paradigm for such systems, enabling efficient communication between learners, educators, and content repositories across geographically distributed environments [1]. The growing number of concurrent users and the heterogeneity of educational resources – including multimedia content, interactive assessments, and collaborative tools – impose stringent requirements on the architectural design of educational platforms.

The relevance of this topic is underscored by the rapid adoption of e-learning solutions, particularly accelerated by global events that have shifted traditional classroom instruction to online environments. Selecting an appropriate architectural approach is critical not only for the performance and scalability of the system but also for ensuring data security, availability, and maintainability over the platform lifecycle [2]. This paper examines and compares the primary architectural patterns applicable to client-server educational platforms.

Architectural Approaches and Their Application in Educational Platforms

Modern client-server systems for educational platforms can be designed following three primary architectural paradigms: monolithic architecture, service-oriented architecture (SOA), and microservices architecture. Each paradigm presents a distinct set of trade-offs relevant to the functional and non-functional requirements of educational systems.

Monolithic Architecture

In a monolithic architecture, all functional components – user management, content delivery, assessment engines, and reporting – are integrated into a single deployable unit. This approach simplifies development and initial deployment, making it suitable for small-scale platforms with a limited user base. However, monolithic systems exhibit significant limitations as platforms scale: any modification to a single module requires redeployment of the entire application, leading to increased downtime and deployment risk [3]. Additionally, horizontal scaling of specific high-demand components, such as video streaming or assessment grading, becomes impractical without scaling the entire system, resulting in inefficient resource utilization.

Service-Oriented Architecture (SOA)

Service-oriented architecture addresses several limitations of monolithic systems by decomposing the application into independently deployable services that communicate through standardized protocols, typically SOAP or REST over HTTP. SOA introduces an enterprise service bus (ESB) to mediate inter-service communication, enabling integration with legacy systems and third-party educational tools [4]. For educational platforms, SOA facilitates the modular integration of learning management systems (LMS), content repositories, and authentication providers. Nevertheless, the ESB introduces a potential single point of failure and may create a performance bottleneck under high concurrency, which is common in examination periods or live lecture streaming scenarios.

Microservices Architecture

The microservices architectural pattern has gained prominence as the preferred approach for large-scale educational platforms due to its emphasis on independent deployability, technology heterogeneity, and fine-grained scalability. Each microservice encapsulates a bounded domain context – for example, authentication, content management, discussion forums, or analytics – and communicates via lightweight APIs, predominantly RESTful HTTP or asynchronous message brokers such as Apache Kafka [5]. This decoupling enables development teams to independently iterate on and scale individual services according to demand.

For an educational content distribution platform, the microservices pattern supports the following key capabilities: (1) independent scaling of content delivery services during peak access periods; (2) isolated fault domains, ensuring that a failure in the assessment service does not affect content browsing; (3) polyglot persistence, allowing each service to utilize the database technology most suited to its data model; and (4) continuous deployment pipelines for individual services, reducing time-to-feature [6].

API Gateway and Security Considerations

Regardless of the chosen architectural style, educational platforms require a unified entry point for client requests. An API gateway pattern fulfills this role by providing request routing, load balancing, authentication enforcement, rate limiting, and SSL termination [7]. In the context of educational data, the gateway also enforces compliance with data protection regulations such as GDPR by centralizing authorization logic and audit logging. The integration of OAuth 2.0 and OpenID Connect protocols is recommended for federated identity management across institutional identity providers.

Cloud Deployment and Scalability

Contemporary educational platforms increasingly leverage cloud infrastructure to achieve elastic scalability. Container orchestration platforms such as Kubernetes enable automated deployment, scaling, and management of microservices, while content delivery networks (CDNs) optimize the distribution of static educational assets to geographically dispersed learners [8]. The adoption of infrastructure-as-code (IaC) practices further enhances reproducibility and disaster recovery capabilities, which are critical for maintaining service availability during peak academic periods.

Conclusions

The selection of an architectural approach for a client-server educational platform must be guided by the specific scalability, reliability, and maintainability requirements of the system. Monolithic architecture remains viable for early-stage platforms with constrained resources, while SOA offers a transitional approach for integrating diverse educational services. Microservices architecture, supported by API gateway patterns and cloud-native deployment strategies, provides the most robust foundation for large-scale educational content distribution and exchange platforms.

Future work should focus on empirical evaluation of architectural trade-offs in terms of latency, throughput, and operational cost within real-world educational deployment scenarios. The integration of event-driven communication patterns and serverless computing components represents a promising direction for further enhancing the responsiveness and cost efficiency of educational platforms.

REFERENCES

1. Sommerville I. Software Engineering. 10th ed. Pearson Education, 2016. 816 p.
2. Richardson C. Microservices Patterns. Manning Publications, 2018. 520 p.

3. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. O'Reilly Media, 2021. 616 p.
4. Erl T. SOA: Principles of Service Design. Prentice Hall, 2008. 608 p.
5. Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media, 2017. 616 p.
6. Burns B., Beda J., Hightower K. Kubernetes: Up and Running. 3rd ed. O'Reilly Media, 2022. 326 p.
7. Geewax J. API Design Patterns. Manning Publications, 2021. 512 p.
8. Hardt D. The OAuth 2.0 Authorization Framework. RFC 6749. Internet Engineering Task Force (IETF), 2012. URL: https://www.rfc-editor.org/rfc/rfc6749 (Accessed: May 25, 2026).

Лавренюк Арсен Олександрович – студент групи 1ПІ-22б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e–mail: arsenlavreniuk@gmail.com.

Майданюк Володимир Павлович – кандидат технічних наук, доцент, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e–mail: maidaniuk2000@gmail.com.

Lavreniuk Arsen O. – student of group 1PI-22b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: arsenlavreniuk@gmail.com.

Maidaniuk Volodymyr P. – Candidate of Technical Sciences, Associate Professor, Associate Professor of the Software Engineering Department, Vinnytsia National Technical University, Vinnytsia, e–mail: maidaniuk2000@gmail.com.