Analysis of methods and models for implementing databases and web interfaces: a case study of student management information systems*

Leonid Romaniuk[1,*,†], Sergey Subbotin[2,†], Ihor Chykhira[1,†], Olesya Shtanyuk[1,†] and Halyna Tulaidan[3,†]

[1] *Ternopil Ivan Puluj National Technical University, 56, Ruska Street, Ternopil 46001,* Ukraine
[2] *National University Zaporizhzhia Polytechnic, Zhukovskogo str., 64, Zaporizhzhia, 69011, Ukraine*
[3] *Ternopil Volodymyr Hnatiuk National Pedagogical University, 2, Maxyma Kryvonosa Street, Ternopil 46027,* Ukraine

Abstract

The article presents an analysis of modern methods and tools for implementing databases and web interfaces in student management systems, covering both traditional and modern approaches to software development. It begins with a review of architectural solutions where Java is used to implement core services, while MySQL ensures data storage with support for ACID transactions, complex queries, triggers, and stored procedures. The application of Django for building RESTful APIs enables the separation of the client-side from internal database operations, facilitating rapid prototyping and automatic database migrations. At the same time, Vue.js provides a dynamic and reactive interface that enhances user interaction with the system. The article also highlights security issues, emphasizing on using JWT for authentication, and the integration of Auth0 facilitates request validation. The advantages of using Spring Boot in enterprise solutions are discussed, where Spring Data JPA reduces boilerplate code when working with relational databases, and configuration files provide MySQL connection settings. Particular attention is paid to the comparison of monolithic and microservice architectures. The study underscores the importance of containerization through Docker, which enables the creation of a stable environment throughout development, testing, and production deployment. The implementation of Infrastructure as Code (IaC) principles using Terraform or Ansible ensures automation in server deployment, load balancing, and network resource management, which is critical for cloud infrastructures. The integration of GraphQL optimizes queries by allowing clients to retrieve only the necessary data. The article also highlights the relevance of classical solutions in C++ with direct MySQL connectivity via the C API. The results of this analysis contribute to well-grounded decision-making in the development of software products focused on information management and user interaction.

Keywords

Endpoints, microservice architecture, blue-green deployment, containerization, token validation, business logic

1. Introduction

Today in the digital environment, it is impossible to develop effective information systems without the proper organization of data storage, processing, and access. Databases and web interfaces are crucial to the software functionality that is used in various industries, starting from education and healthcare to finance and industry. With the growing demands for security and speed of system deployment, developers are increasingly implementing projects using combined technological solutions that incorporate different tools and architectural approaches.

The choice of an appropriate database model and interface creation tools depends not only on specific tasks, but also on the extent to which the system must support integration with other

AdvAIT-2025: 2nd International Workshop on Advanced Applied Information Technologies: AI & DSS, December 05, 2025, Khmelnytskyi, Ukraine, Zilina, Slovakia
[1*] Corresponding author.
[†] These authors contributed equally.

leonidromanyuk@ukr.net (L. Romaniuk); subbotin@zntu.edu.ua (S. Subbotin); ig.vi.chi@gmail.com (I. Chykhira); olesyashtanyuk@ukr.net (O. Shtanyuk); tulaidan@tnpu.edu.ua (H. Tulaidan)
0000-0002-2538-4026 (L. Romaniuk); 0000-0001-5814-8268 (S. Subbotin); 0000-0002-8615-3635 (I. Chykhira); 0000-0002-4388-2942 (O. Shtanyuk); 0000-0003-2306-6435 (H. Tulaidan)

services, adapt to changes in the data structure, or ensure the protection of users' personal information. In this context, it is necessary to conduct an in-depth analysis of modern methods and means of implementation, which allows us to develop an idea of the most effective solutions for different usage scenarios.

The relevance of the study is driven by the need for practical recommendations for developers, software architects and system integrators who are searching for optimal approaches to the implementation of information systems, taking into account real-world constraints, standards, market trends and technological progress.

2. Analysis of recent research and publications

To begin with, the paper [1] discusses the methodology for installing PHP extensions in Docker images without using the default disabled PECL, which is especially relevant with PHP 7.4. It demonstrates a step-by-step approach using tools such as docker-php-ext-configure and docker-php-ext-install that allow creating clean and modular Docker files tailored to specific project requirements. Practical instructions include the APCu, Redis, Igbinary, and MongoDB extensions. Particular attention is paid to the problems that arise when installing some extensions, in particular MongoDB, which requires working with submodules and multi-stage builds.

The paper [2] compares the performance of three PHP frameworks — Laravel, Symfony, and CodeIgniter. The analysis is based on the Model-View-Controller (MVC) architecture and the application of the QSOS evaluation methodology, which allows measuring the number of requests per second, memory usage, response time, and other parameters. The results show that Laravel is dominant due to its high performance, low response time, and ability to handle a large number of requests. In turn, Symfony, although multifunctional, requires deeper knowledge in order to implement complex solutions, while CodeIgniter, due to its simplicity, is optimal for small and medium-sized projects with simpler requirements.

Research [3] focuses on the development of a web-based Docker Image Assistant Generator (DIAG) tool in the context of User-PC Computing system (UPC). DIAG adopts Angular for creating a dynamic interface, Laravel for handling the server logic and data processing using RestAPI, MySQL for storing structured data, and Shell scripting in order to automate processes. A key feature of the tool is the ability to modify the source code directly during the performance of a task, which allows users to update Docker images in real time. The use of the MVC pattern facilitates the separation of tasks.

Research [4] analyses the tools for deploying and managing cloud resources — Google Cloud Deployment Manager and Terraform. The assessment is based on testing the resource deactivation time, which allows determining the operational efficiency of these technologies. The results show that Terraform provides faster resource deactivation due to its independence from specific cloud platforms. However, Google Cloud Deployment Manager has better native integration with Google Cloud resources, which can be a decisive factor for projects that operate exclusively within GCP.

Paper [5] investigates the impact of microservice architecture and containerisation on resource efficiency and overall performance of SAAS CRM systems. Experiments proved that the use of microservices in a containerised environment, in particular, using Docker and orchestration with Kubernetes, can significantly reduce CPU and memory consumption compared to monolithic systems.

Thus, the analysis of the literature shows a wide range of modern technological solutions for optimising software development. However, taking into account the above-mentioned scientific publications, the issue related to the development of database and web interface implementation infrastructure remains still insufficiently studied and requires further elaboration.

3. Statement of the task

The aim of the research is to review the methods and models of implementing databases and web interfaces on the example of various student management systems.

4. Presentation of the main material

The present study examines a range of contemporary implementations of student management systems and analyses their operational principles, technological composition, and architectural characteristics. Such systems increasingly serve as central components of modern educational infrastructures, requiring high reliability, configurability, and scalability. The discussion below systematises five principal implementation approaches, highlighting their software stacks, integration mechanisms, and security considerations. To provide analytical clarity, each subsection concludes with a summarised Results component that emphasises the main findings associated with the corresponding technological approach.

1. Java-Based Backend with Django, Vue.js, and MySQL Integration.

One of the more hybridised implementations of student education management systems combines a Java backend with MySQL-based data persistence and a layered presentation strategy utilising both Django and Vue.js [6].

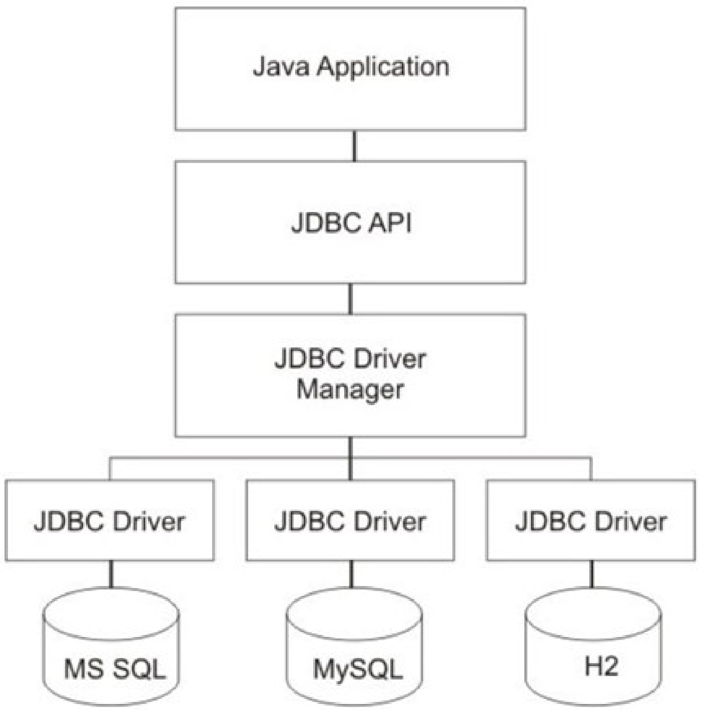


Figure 1: Connecting a Java application to a database.

In this configuration, Java functions as the foundation for the core business logic, benefiting from its maturity, threading model, and ability to maintain high throughput under concurrent workloads. MySQL, as the relational database component, contributes ACID-compliant transactions, stored procedures for encapsulating database logic, triggers for reactive updates, and robust JOIN capabilities essential for querying interconnected datasets [7].

A notable feature of this approach is the use of parameterised Java queries, which enhance protection against SQL injection attacks by separating query logic from user-provided values. Django, operating above the Java layer, offers RESTful endpoints that mediate interactions between client-side components and internal services.

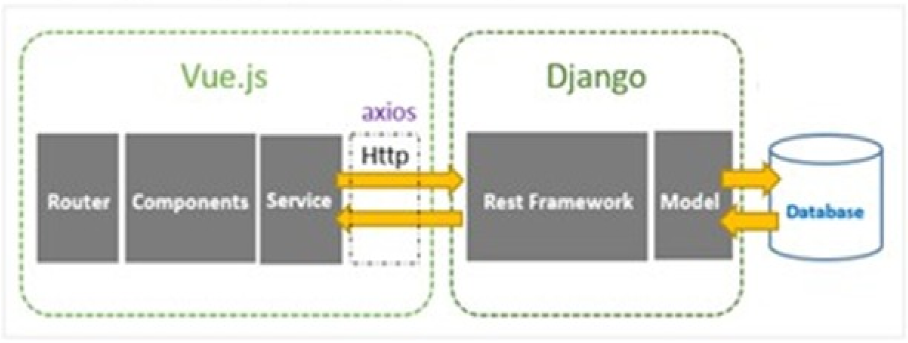


Figure 2: Vue.js relationship with Django.

This separation ensures that the client layer remains unaware of the underlying database structure, improving maintainability and facilitating the introduction of additional security layers.

From the frontend perspective, Vue.js provides reactive rendering and efficient state management via asynchronous communication with the Django REST API. Libraries such as Axios or the native fetch API enable smooth data exchange between interface components and backend services. Additionally, authentication procedures can be reinforced using Auth0, which validates tokens via JSON Web Key Sets retrieved from trusted endpoints.

2. Spring Boot and MySQL with Structured Configuration.

For enterprise-oriented development, Spring Boot represents a more consolidated Java-centred approach. Its configuration model, typically expressed through application.properties or application.yml files, provides an automated and structured environment for linking the application with a MySQL database. Through Spring Data JPA, repository interfaces encapsulate database operations, drastically reducing boilerplate code and allowing developers to express complex queries declaratively [8].

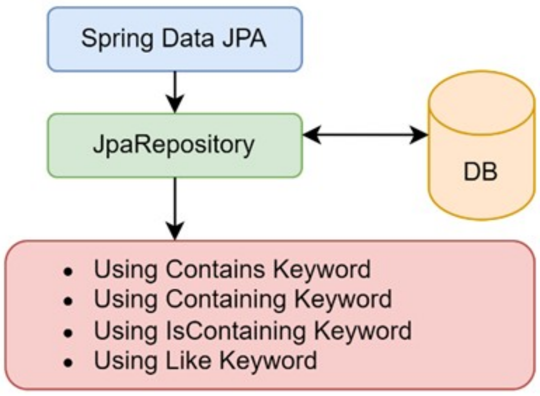


Figure 3: Using JPA Repository to search for records.

REST controllers built within Spring Boot map HTTP requests to service-layer operations, providing a unified interface across multiple frontends. The business logic layer ensures transactional integrity, especially when coordinating interactions across several repositories. Because configurations, dependencies, and ORM mappings are largely automated, Spring Boot supports a high degree of scalability and reproducibility.

3. Rapid Prototyping via Django and Vue.js with Automated Migrations.

For environments requiring rapid iteration, Django paired with Vue.js constitutes a productive development model. Django's ORM allows developers to design domain-specific models that are automatically translated into SQL schema definitions during migrations. This autonomous process minimises the initial database design workload and facilitates fast revisions during the early stages of system prototyping.

Django's templating engine assists in generating server-rendered HTML layouts, into which Vue.js components can be dynamically mounted [9]. This hybrid rendering model enables a smooth transition from traditional server-side page generation to fully interactive client-side interfaces. In more complex setups, Vuex centralises state management across components, improving consistency in data handling.

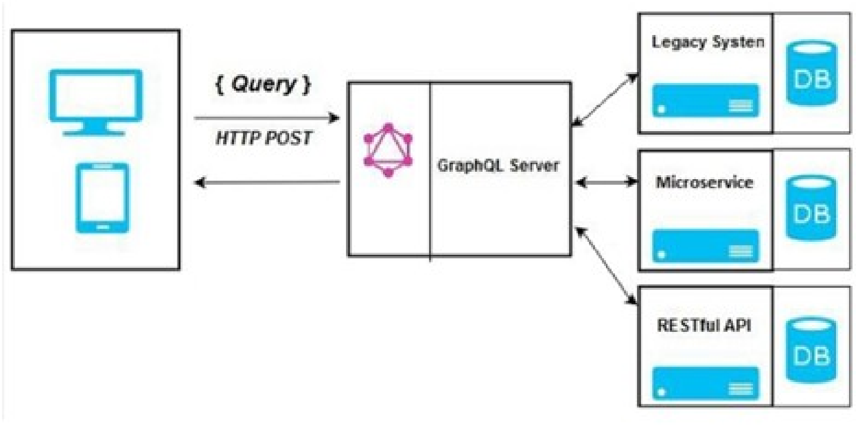


Figure 4: GraphQL client–server architecture.

4. GraphQL-Based Microservice Architecture with Spring Boot and MongoDB.

As data complexity increases, GraphQL has emerged as an attractive alternative to traditional REST architectures. GraphQL allows clients to specify precisely which fields are required, minimising overfetching or underfetching of information [10]. When deployed using Spring Boot in combination with MongoDB, the backend attains flexibility in accommodating frequently changing data structures, as MongoDB's schema-less document model permits structural variation.

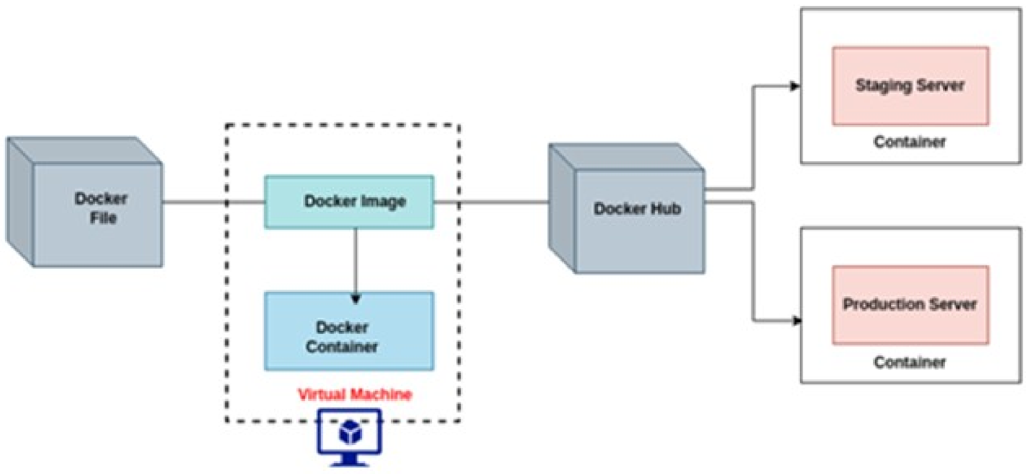


Figure 5: Docker containerization platform.

Resolvers serve as the bridging logic between GraphQL queries and backend services. These resolvers operate according to predefined schemas and allow both highly granular and aggregate queries through a single endpoint. In distributed environments, this architecture supports the unification of disparate microservices under a common API layer.

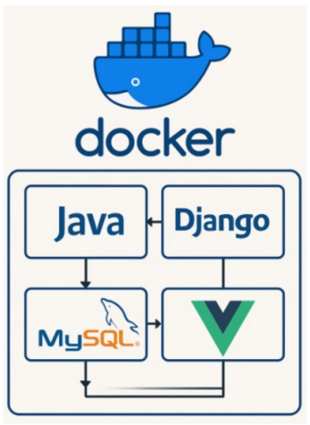


Figure 6: Example of multi-container environment with services for the main application, databases, and optional tools.

5. Classical Implementation Using C++ with MySQL Integration.

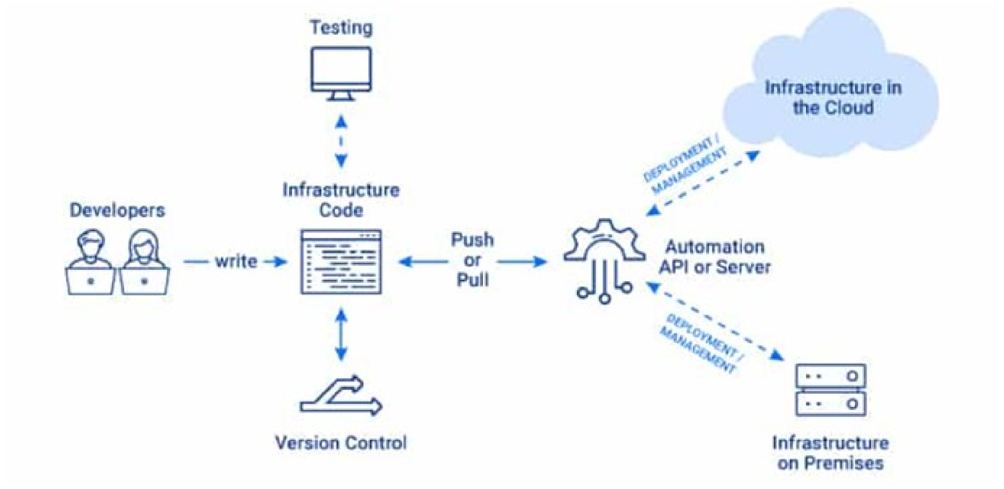
Although less common in modern web-oriented systems, C++ continues to offer a high-performance alternative for developing student management platforms. Its low-level resource management enables fine-grained control over memory and networking operations. Integration with MySQL is facilitated through the MySQL C API, enabling direct execution of database queries [11].

However, the absence of built-in abstractions analogous to ORM frameworks requires developers to manually handle data transformations, memory allocation, and security checks. While this approach yields performance advantages, it also increases development complexity and extends the learning curve for new contributors.

Containerisation and DevOps Considerations.

Irrespective of the underlying implementation, containerisation plays a central role in modern deployment workflows. Docker ensures that applications run consistently across development, testing, and production stages by isolating dependencies within container images. Multi-container environments commonly include application servers, databases, message brokers, and auxiliary utilities. When scaling becomes necessary, orchestration tools allow replication, rolling updates, and blue–green or canary deployment strategies.

Figure 7:



Integration Infrastructure as Code (IaC).

Infrastructure as Code (IaC) techniques using Terraform or Ansible further support consistent provisioning of cloud environments. By codifying network configurations, load balancers, and server parameters, IaC ensures reproducibility and enables controlled incremental rollouts [12].

Future implementations are likely to benefit from frameworks such as FastAPI, which supports asynchronous operations and integrates effectively with SQLAlchemy. TypeScript-based backends

(NestJS or Express) offer unified development experiences across client- and server-side logic. Furthermore, advancements in machine learning, probabilistic models, and neural network–based pattern recognition may significantly enhance student analytics, document processing, and personalised feedback mechanisms.

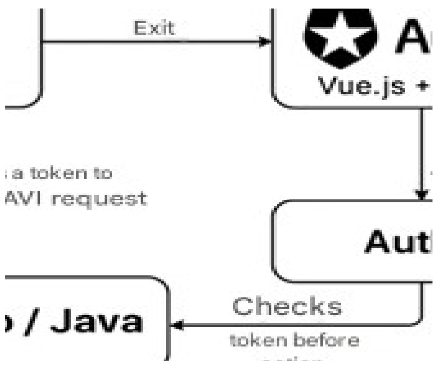


Figure 8: Auth0 authentication sequence in Vue.js frontend with Django/Java backends.

A consistent authentication model is essential when coordinating multiple frameworks or microservices. Java backends typically validate JSON Web Tokens (JWTs), while Django leverages middleware and decorators to restrict access to protected views. Vue.js components can dynamically adjust rendered elements based on token presence or session validity. GraphQL implementations apply token checks directly within resolvers. Cross-service authentication can be unified using Auth0 or similar providers that issue signed tokens recognisable by all participating services.

5. Results&Discussion

Approach 1

- The hybrid stack benefits from modularity but introduces additional integration overhead between Java, Django, and Vue.js.
- Security is strengthened through layered token validation, parameterised database queries, and Django middleware.
- The architecture offers rapid frontend response and clear separation of concerns, though its operational complexity increases proportionally with system scale.

Approach 2

- JPA repositories significantly streamline data access logic and reduce development time.
- The approach is well-suited for both monolithic and microservice deployments due to its standardised configurations.
- Security based on JWT and role-based access policies integrates naturally with Spring's authentication mechanisms.

Approach 3

- This approach enables rapid prototyping due to automated schema management and flexible template rendering.
- It reduces initial development costs but may face limitations when handling heavy traffic or high concurrency.
- Authentication is readily available through Django's built-in systems, enabling secure trial deployments.

Table 1
Key aspects of the described approaches

Approach	Backend	Frontend/API	Security
№ 1 – Java + Django & Vue.js + MySQL with ACID support	Java for basic services, Django for REST API	Django for server-side technologies (HTML generation, REST API); Vue.js for client response (fetch/Axios)	JWT validation; Auth0 for token validation
№ 2 – Spring Boot + MySQL with configuration files	Spring Data JPA	REST controllers; integration with each frontend	JWT and role validation
№ 3 – Rapid prototyping (Django + Vue.js) + MySQL with automigrations	Django with ORM	Vue.js for server-generated pages	Built-in Django authentication and JWT
№ 4 – Microservice architecture (GraphQL з Spring Boot and MongoDB)	Spring Boot with GraphQL resolvers	Single endpoint for all data	Tokenized checks for resolvers
№ 5 – C++ + MySQL via C API	Pure C++ for low-level control	UI generation occurs directly	Manual security and memory management

Approach 4

- GraphQL ensures efficient client-side data retrieval by eliminating unnecessary fields.
- MongoDB complements this approach through schema flexibility suited for evolving datasets.
- The architecture supports microservice models, though it increases the need for robust service discovery and debugging mechanisms.

Approach 5

- C++ provides superior performance but lacks automation and abstraction layers typical of modern frameworks.
- Security and memory management must be implemented manually, increasing the likelihood of implementation errors.
- The approach is suitable for specialised systems but is less practical for large-scale or rapidly evolving educational platforms.

General Results.

1. Modern student management systems rely on multi-layered architectural patterns that balance performance, flexibility, and maintainability.
2. Hybrid implementations provide strong modularity but introduce operational complexity.
3. Spring Boot offers an enterprise-grade framework with automated configuration and scalable deployment options.
4. Django–Vue.js combinations are optimal for prototyping.
5. GraphQL-based microservices support fine-grained queries and flexible modelling.
6. C++ solutions offer high performance but increased development cost.
7. Containerisation and IaC underpin modern deployment reliability.

6. Conclusions

In general, a comprehensive analysis of the methods and means of implementing databases and web interfaces in the context of student management systems allows us to identify both the strengths and limitations of each approach depending on the context of application. The combination of Java or Spring Boot with MySQL and modern frontend technologies such as Vue.js ensures high stability and security, while Django simplifies prototyping and speeds up development through process automation. The application of machine learning for predictive analytics in data backup strategies can significantly improve the reliability and security of student management systems. Innovations in hardware and communication technologies, such as advanced antenna systems, can support the development of more efficient and scalable educational information infrastructures.

The integration of GraphQL, Docker containerization, and IaC implementation contribute to the creation of a scalable architecture. Consideration of classic programming languages such as C++ demonstrates the continuing relevance of low-level solutions in specific cases.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] O. Laviale, Installing PHP extensions from source in your Dockerfile (2019). URL: https://olvlvl.com/2019-06-install-php-ext-source.html

[2] M. Laaziri, K. Benmoussa, S. Khoulji, L. Kerkeb, A comparative study of PHP frameworks performance, Procedia Manuf. 32 (2019) 864–871. doi:10.1016/j.promfg.2019.02.295.

[3] L. Htet Aung, N. Funabiki, S. Aung, X. Zhou, X. Xiang, W.-C. Kao, A web-based Docker image assistant generation tool for user-PC computing system, Information 14 (2023) 300. doi:10.3390/info14060300.

[4] S. Chinthapatla, Unleashing the power of AWS: revolutionizing cloud management through infrastructure as code (IaC) (2024).

[5] M. Röser, Customer relationship management in new business models (2024). doi:10.5772/intechopen.114840.

[6] A. Bouchefra, Building modern applications with Django and Vue.js, Auth0 Blog (2021). URL: https://auth0.com/blog/building-modern-applications-with-django-and-vuejs/

[7] S. Trimal, Z. Shaikh, A. Chavan, S. M. Kamble, Student management system, JournalNX 10(6) (2024) 116–118.

[8] Waqasafzal, Integrating MySQL with Spring Boot: An in-depth student management example, Medium (2024). URL: https://medium.com/@waqasafzal/integrating-mysql-with-spring-boot-an-in-depth-student-management-example-6c2de0d18b06

[9] L. del Alba, How to prototype a web app with Django and Vue.js, SitePoint (2020, updated 2024). URL: https://www.sitepoint.com/prototype-web-app-django-vuejs/

[10] A. Jhingran, Accessing SQL and NoSQL databases with a GraphQL API, StepZen Blog (2021). URL: https://stepzen.com/blog/accessing-sql-and-nosql-databases-with-a-graphql-api

[11] S. Zhong, M. Rigger, Understanding and reusing test suites across database systems, Proc. ACM Manag. Data 2(6) (2024) Article 253. doi:10.1145/3698829.

[12] M. Eleraky, W. Anis Aziz, J. Soliman, Using cloud infrastructure as a code (IaC) to set up web applications, Int. J. Simul.: Syst., Sci. Technol. 24 (2023).