One-gateway system in managing campus information system using microservices architecture

Yulita Salim [a,1], Ismunandar Muis [a,2,*], Lukman Syafie [a,3], Huzain Azis [a,4], Abdul Rachman Manga [a,5]

[a] Universitas Muslim Indonesia, Makassar, Indonesia
[1] yulita.salim@umi.ac.id; [2] indarmuis.labfik@umi.ac.id; [3] lukman.syafie@umi.ac.id; [4] huzain.azis@umi.ac.id; [5] abdulrachman.manga@umi.ac.id
* corresponding author

ARTICLE INFO

Article history
Received July 11, 2023
Revised July 30, 2023
Accepted November 24,2023

Keywords
Information system
Microservice
Development
Application
Architecture

ABSTRACT

Universitas Muslim Indonesia (UMI) has developed several applications for managing the campus's digital information and management systems, both internally and externally. However, several applications were previously created in the development of information system applications at UMI. However, these applications were not well-suited for long-term use due to their complexity and lack of integration. Therefore, UMI aims to create a fully integrated and well-managed campus information system by implementing the concept of microservices. The microservices approach involves dividing large applications into smaller interconnected components. This approach facilitates the management of application systems and enables better integration. Moreover, the microservices approach simplifies system maintenance for application developers, as each application is separated into smaller components.

1. Introduction

In the ongoing digital era, developing applications capable of managing information systems has become vital for institutions or companies. Practical information systems have significantly benefitted institutions or companies that have digitized most of their internal and external activities to meet these challenges. Information technology and electronic communication have enhanced the potential for information exchange within campuses [1]. However, with the rapid progress in information system development, the need for more complex applications with higher scalability is growing [2].

The selection of the appropriate architecture in software development is crucial for organizing code, managing business logic, and ensuring applications can be effectively developed, tested, and maintained. The chosen architecture also influences the flexibility for future changes and application development. In this context, various types of architecture are used in software development, such as monolithic architecture, service-oriented architecture (SOA), object-oriented architecture, event-driven architecture, and microservices architecture. Each architecture type has different characteristics and design principles, and the selection of the suitable architecture depends heavily on the needs and objectives of the application being developed.

In the ongoing development of a campus information system, selecting microservices architecture is a relevant and appropriate step. Microservices architecture is a software development approach where the overall software functionality is provided by smaller software components known as microservices [3], [4]. Each microservice in this architecture focuses on a specific task or function, allowing for better development and periodic system testing. This approach breaks down an extensive application into separate independent applications or services, making managing and maintaining complex campus information systems easier.

In implementing microservices architecture for managing a campus information system, one key aspect is utilizing a one-gateway system. A one-gateway system is an approach where a single gateway controls access and connections between different microservices within the system. By leveraging a one-gateway system, users can access various services through a single access point, enhancing efficiency and user convenience when interacting with the campus information system. Moreover, this approach enables efficient access management and request distribution.

Implementing microservices architecture utilizing a one-gateway system in managing a campus information system brings significant advantages. This approach allows for better scalability, where microservice components can be added or reduced independently as needed. Secondly, system maintenance and development become easier and separated between each service. Thirdly, microservices architecture enables development teams to work independently on each microservice without disturbing other system parts [5]. Fourthly, with a one-gateway system, users can easily access various services through a single access point, improving efficiency and providing a better user experience.

Overall, implementing microservices architecture with the utilization of a one-gateway system in managing a campus information system is a suitable and effective step in addressing the complexity and scalability faced by educational institutions today. This approach enables more effective, efficient, and well-organized development of campus information systems, offering long-term benefits in application development and maintenance. By adopting microservices architecture and leveraging a one-gateway system, campus information systems can adapt to evolving changes and needs while delivering better services to users.

2. Method

The current development of the campus information system will employ the microservices architecture approach, as illustrated in (Fig. 1).

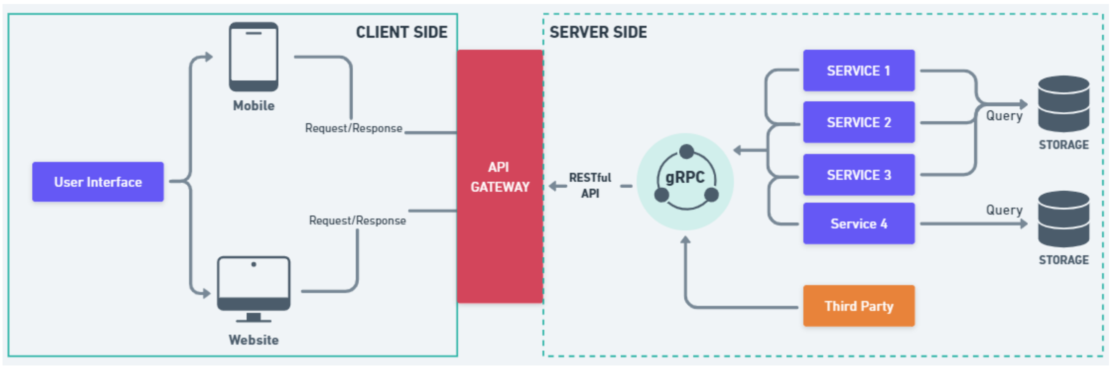


Fig. 1. Microservice Architecture

The microservices architecture consists of several main components. The User Interface (UI) directly interacts with users. The API Gateway manages access to the microservices and ensures security. gRPC is an efficient communication protocol used between services. The microservices perform specific tasks independently. Third-party services integrate external services. The Storage database stores and manages data [6], [7]. By combining these components, the microservices architecture allows for the development a distributed, modular system that can be rapidly customized.

2.1. Technology Used

The following are several technologies employed during the development of the campus information system using microservices architecture :

- Design Tools

The design tool utilized for application development is Whimsical. Whimsical is a visual design tool or platform used to create various types of diagrams or wireframes in an easily comprehensible manner. Using Whimsical, users can depict ideas, design user interfaces, create flowcharts, and generate other visual representations in software development projects, product design, or business process modeling.

- Front – End

  Front-end is a crucial part of software development that directly interacts with users. In this particular development, two Front-End technologies are utilized for the system: Website and Mobile Application. Here are the explanations for each.

  - Flutter

    Flutter is a cross-platform application development framework developed by Google. Using the Dart programming language, Flutter is designed to create rich and responsive user interfaces for mobile, web, desktop, and embedded devices. To better understand how Flutter works, refer to (Fig. 2).

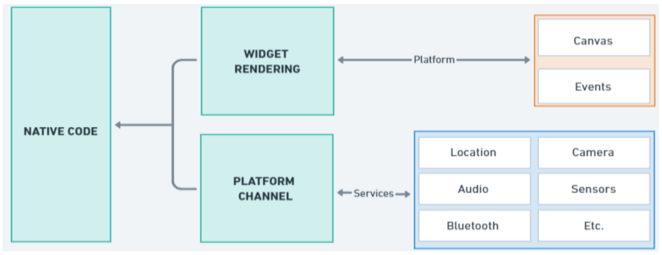


Fig. 2. Flutter Framework Development Stages

(Fig. 2) illustrates the functioning of Flutter and the essential concepts that support application performance. Native Code is employed for specific features unavailable within the Flutter framework. Widget Rendering transforms visual components into screen displays. The Platform Channel enables communication between Flutter and Native Code. Services are background components responsible for managing network and data storage. The Platform Canvas is used to render views on the target platform. Events encompass user interactions facilitating real-time responsiveness [8], [9]. Familiarizing oneself with these concepts empowers Flutter developers to build responsive, high-quality applications.

  - Angular

    Angular is an open-source web development framework developed by Google. Utilizing the TypeScript programming language, Angular empowers developers to create complex and dynamic web applications effortlessly. In this system development, Angular is utilized for constructing the website's interface. For a clearer understanding of how Angular operates, refer to (Fig. 3).

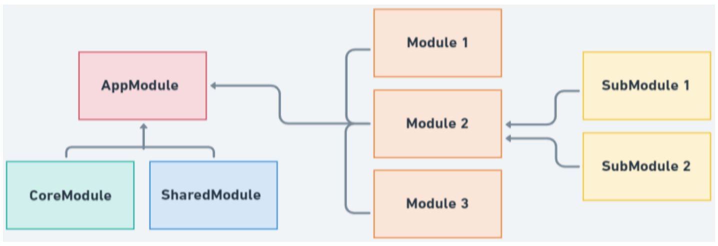


Fig. 3. Angular Component Modules

(Fig. 3) illustrates the architecture of Angular, which consists of several vital components. The app Module is the primary root module that connects all the components, services, and modules within the application. The Core Module contains the core application code, including globally-used services and components. Shared Module functions as a module that houses reusable components, services, and directives across different parts of the application. Modules and Submodules are separate units that organize related functionalities [10]. These concepts facilitate Angular applications' structured, modular, and easily maintainable development.

- Backend Framework

  - Golang Fiber

Golang Fiber is a lightweight and fast web framework for the Go programming language. It provides robust routing, middleware, and validation features to develop responsive web applications. Golang Fiber enables developers to create high-performance APIs and web applications quickly.

- gRPC

gRPC is a Remote Procedure Call (RPC) system developed by Google. It allows different applications to communicate with each other through remote procedure calls. gRPC utilizes an efficient communication protocol and supports various programming languages [11], [12]. With gRPC, developers can define messages and services in protocol buffer files, automatically generating code for inter-application communication [13]. To comprehend how gRPC operates, refer to (Fig. 4).

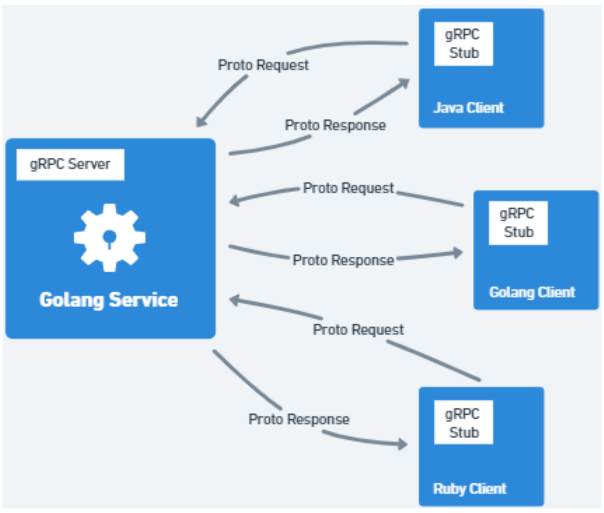


Fig. 4. gRPC Method Workflow

As shown in (Fig. 4), in the gRPC architecture, the "Service" serves as the contract that defines the available operations on the server, while the "Stub" represents the client and is used to communicate with the server. The service functions as an interface that defines the methods callable by the client, while the stub provides locally callable methods for the client. When the client calls a method on the stub, the stub packages the request, sends it to the server over the network, and receives the response from the server [14]. Through the service and stub collaboration, gRPC enables efficient and reliable communication between the client and server.

- Swagger

Swagger is a framework for defining, building, and documenting RESTful APIs. It utilizes the OpenAPI specification to describe APIs, including endpoints, parameters, and responses. Swagger assists developers in generating structured and interactive documentation for APIs built with Golang Fiber and gRPC. The OpenAPI structure of Swagger can be seen in (Fig. 5).

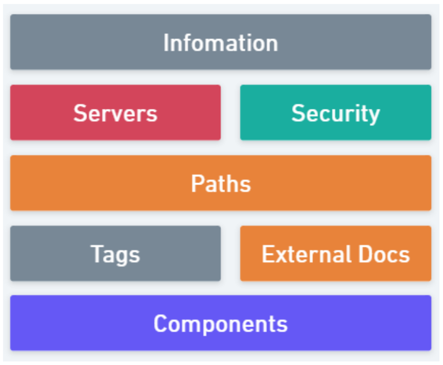


Fig. 5. Swagger OpenAPI Format

- Version Control Ssytem
  - Git

    Git is a widely used and versatile version control system. It serves the purpose of tracking changes in the source code of software projects. Git allows development teams to collaborate effectively and manage source code efficiently. Within Git, repositories store all the code versions, enabling developers to perform actions such as branching, merging changes, tagging versions, and managing the history of changes [15].
  - Gitlab

    GitLab is a software development platform that utilizes Git as its primary version control system. It provides a user-friendly web interface that facilitates developers' interaction with Git repositories. In addition, GitLab offers various advanced features such as project lifecycle management, issue tracking systems, CI/CD pipelines, and user permission management [16]. GitLab can be deployed on a local server or utilized as a cloud-based service through GitLab.com, enabling seamless collaboration among development teams and efficient project management.

2.2. Development Phases

In this system development, Agile is utilized as the development methodology. Agile is a flexible and collaborative approach to software development, as depicted in (Fig. 6).

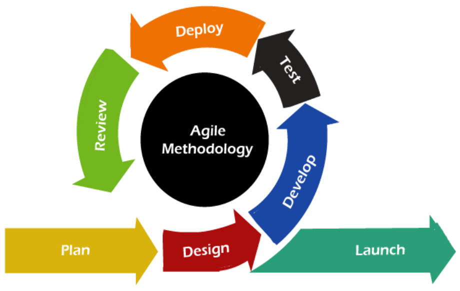


Fig. 6. Agile Software Development Life Cycle

The agile software development approach emphasizes adaptability, collaboration, and rapid iterations. In Agile Development, development progresses incrementally, focusing on delivering quick business value and responsiveness to changes. Teams work in short cycles called sprints, where they continuously identify requirements, plan, develop, and test software [17]–[20]. Here are several Agile phases.

- Planning and Analysis

  The development team collaborates with stakeholders to understand project requirements and objectives. They create a list of requirements and user stories that define the necessary features or functions.
- Sprint Achievement

  Work is planned and divided into short iterations called sprints. The team selects user stories from the backlog (list of requirements) to be worked on during the sprint for several weeks.
- Design and Development

  The development team designs solutions and develops software based on the selected user stories. This process involves prototyping, testing, and iterative refinement to ensure the quality and suitability of the developed solution.
- Testing and Integration

  Once development is completed, testing is conducted to verify the quality of the software. This includes functionality, integration, and performance testing to ensure proper functionality and alignment with user needs.

- Evaluation and Adjustment

  After each sprint, the team and stakeholders conduct evaluations to measure success and improve the development process. Retrospective discussions are held to identify what worked well, areas for improvement, and how to enhance the overall process

- Selection and Planning of the Next Sprint

  Based on retrospective findings and stakeholder feedback, the team selects items from the product backlog for development in the next sprint.

3. Results and Discussion

Microservices architecture is an innovative approach to designing and managing information systems, including in the context of campus information system management. In implementing the one-gateway system, the user interface in the form of mobile applications and websites serves as the primary channels for users to interact with the system. The architecture diagram of the microservices can be seen in (Fig. 7).

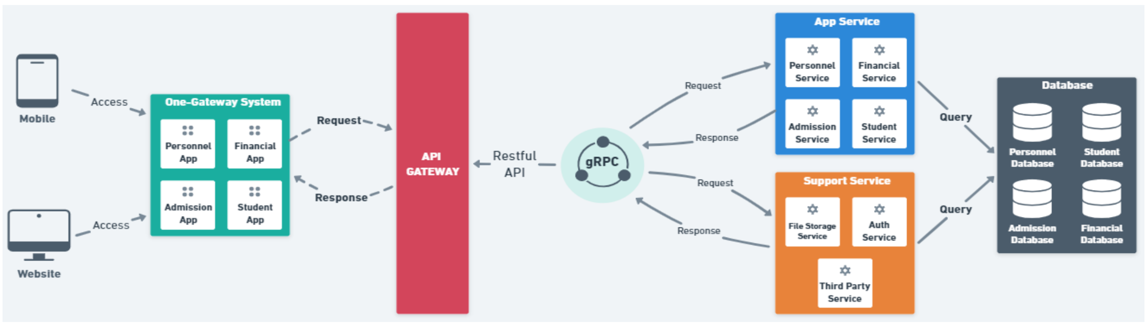
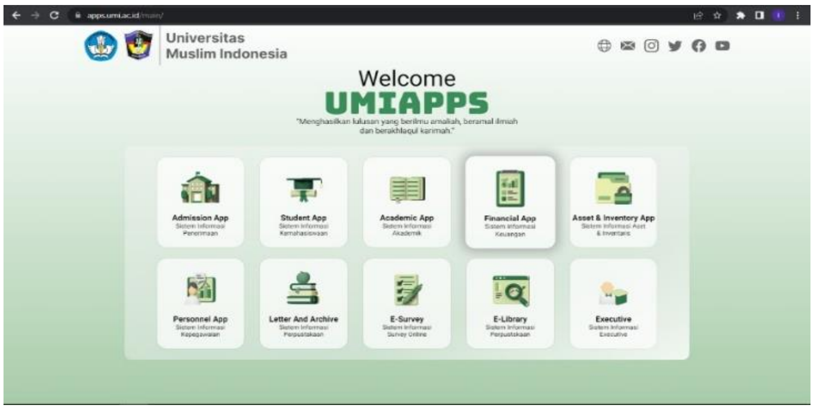


Fig. 7. Campus Information System Using Microservice Architecture

Each microservice within the microservices architecture has separate responsibilities and provides specific functionalities. For example, there are services such as Personnel Services, Financial Services, and other relevant services to the campus information system. Each service requests and receives responses from other services using the gRPC protocol. The relationships between these services in the microservices architecture are also depicted in (Fig. 7). Furthermore, each microservice interacts with a database to access and manipulate the required data. Each service has limited access to the database based on its responsibilities and functionalities. The campus information system can be organized into separate and well-managed components by employing the microservices architecture, enhancing flexibility, scalability, and overall system reliability. In implementing the microservices architecture for the campus information system, the user interface can be accessed through a website developed using Angular and a mobile application built with Flutter. For the website implementation, the user interface is developed using the Angular framework, which enables the creation of responsive and interactive user interfaces. (Fig. 8) illustrates the architecture implementation on the website using Angular.


apps.umi.ac.id/main/

Universitas Muslim Indonesia

Welcome
UMIAPPS
"Menghasilkan lulusan yang berilmu amaliah, beramal ilmiah dan berakhlaqul karimah."

Admission App — Sistem Informasi Penerimaan
Student App — Sistem Informasi Kemahasiswaan
Academic App — Sistem Informasi Akademik
Financial App — Sistem Informasi Keuangan
Asset & Inventory App — Sistem Informasi Aset & Inventaris
Personnel App — Sistem Informasi Kepegawaian
Letter And Archive — Sistem Informasi Perpustakaan
E-Survey — Sistem Informasi Survey Online
E-Library — Sistem Informasi Perpustakaan
Executive — Sistem Informasi Executive

Fig. 8. Interface Website Implementation

In addition to the website, the campus information system can be accessed through a mobile application developed using the Flutter framework. Flutter allows for the rapid development of cross-platform mobile applications for both Android and iOS. (Fig. 9) depicts the architecture implementation for the mobile application using Flutter.

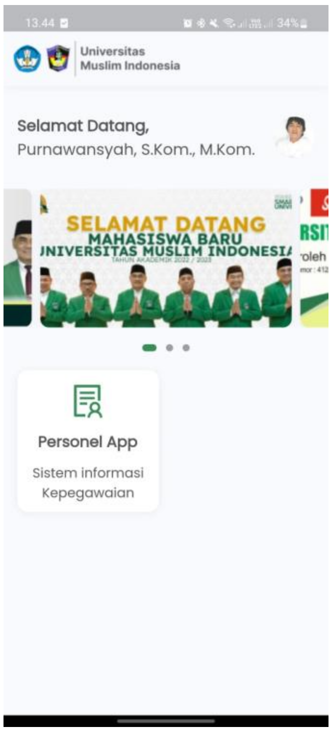

Universitas Muslim Indonesia
Selamat Datang,
Purnawansyah, S.Kom., M.Kom.
SELAMAT DATANG
MAHASISWA BARU
UNIVERSITAS MUSLIM INDONESIA
Personel App
Sistem informasi Kepegawaian

Fig. 9. Interface Mobile Implementation

(Fig. 9) illustrates how the user interface in the mobile application interacts with the one-gateway system through the API Gateway. Users perform operations or requests through the mobile application, which are sent through the API Gateway to the relevant microservices. The services then query the database to access and manipulate the required data. The service responses are sent back through the API Gateway and displayed on the user interface in the mobile application.

The campus information system can be accessed through various platforms with customized user interfaces by utilizing Angular for the website and Flutter for the mobile application implementation. The microservices architecture enables seamless communication between these components, with the one-gateway system serving as the single entry point. As a result, users can easily access and interact with the campus information system through the website and mobile application according to their preferences and needs.

4. Conclusion

Based on the system development conducted, implementing the One-Gateway System using a microservices architecture has proven effective and efficient in managing the campus information system. This approach utilizes a single entry point (One-Gateway System) to integrate and manage various microservices within the campus environment. The microservices architecture can build the campus information system with separate and independent components, providing better scalability, flexibility, and scalability. The One-Gateway System is crucial in connecting users to these services, managing authentication and authorization, and providing a uniform and user-friendly interface. Implementing the One-Gateway System brings significant benefits, such as improved security with better access control, reduced complexity in service management, and increased productivity in developing and maintaining the campus information system. However, challenges such as gateway system scalability, coordination among development teams, and management of microservice versioning need to be carefully addressed and handled through further research and development.

Implementing the One-Gateway System with a microservices architecture offers a promising approach to managing the campus information system. This article provides a strong foundation for developers and IT professionals to design and implement an efficient and integrated campus information system. By leveraging the significant benefits offered by this approach and carefully addressing the existing challenges, educational institutions can obtain optimal advantages from their information technology infrastructure.

References

[1] R. Malhotra, D. Kumar, and D. P. Gupta, “An android application for campus information system,” *Procedia Comput. Sci.*, vol. 172, pp. 863–868, 2020, doi: 10.1016/j.procs.2020.05.124.

[2] A. Cohen, “Analysis of student activity in web-supported courses as a tool for predicting dropout,” *Educ. Technol. Res. Dev.*, vol. 65, no. 5, pp. 1285–1304, Oct. 2017, doi: 10.1007/s11423-017-9524-3.

[3] K. Bakshi, “Microservices-based software architecture and approaches,” in *2017 IEEE Aerospace Conference*, Mar. 2017, pp. 1–8, doi: 10.1109/AERO.2017.7943959.

[4] O. Zimmermann, “Microservices tenets,” *Comput. Sci. - Res. Dev.*, vol. 32, no. 3–4, pp. 301–310, Jul. 2017, doi: 10.1007/s00450-016-0337-0.

[5] F. Freitas, A. Ferreira, and J. Cunha, “A methodology for refactoring ORM-based monolithic web applications into microservices,” *J. Comput. Lang.*, vol. 75, no. April, p. 101205, 2023, doi: 10.1016/j.cola.2023.101205.

[6] H. Suryotrisongko, D. P. Jayanto, and A. Tjahyanto, “Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot,” *Procedia Comput. Sci.*, vol. 124, pp. 736–743, 2017, doi: 10.1016/j.procs.2017.12.212.

[7] A. Pavlenko, N. Askarbekuly, S. Megha, and M. Mazzara, “Micro-frontends: Application of microservices to web front-ends,” *J. Internet Serv. Inf. Secur.*, vol. 10, no. 2, pp. 49–66, 2020. [Online]. Available at : https://jisis.org/wp-content/uploads/2022/11/jisis-2020-vol10-no2-04.pdf.

[8] O. M. A. AL-atraqchi, “A Proposed Model for Build a Secure Restful API to Connect between Server Side and Mobile Application Using Laravel Framework with Flutter Toolkits,” *Cihan Univ. Sci. J.*, vol. 6, no. 2, pp. 28–35, 2022, doi: 10.24086/cuesj.v6n2y2022.pp28-35.

[9] N. Simiriotis and R. Palacios, “A Flutter Prediction Framework in the Open-Source SU2 Suite,” *AIAA Sci. Technol. Forum Expo. AIAA SciTech Forum 2022*, pp. 1–15, 2022, doi: 10.2514/6.2022-1327.

[10] K. Bielak, B. Borek, and M. Plechawska-Wójcik, “Web application performance analysis using Angular, React and Vue.js frameworks,” *J. Comput. Sci. Inst.*, vol. 23, no. November 2021, pp. 77–83, 2022, doi: 10.35784/jcsi.2827.

[11] C. Nimpattanavong, I. Khan, T. Van Nguyen, R. Thawonmas, W. Choensawat, and K. Sookhanaphibarn, “Improving Data Transfer Efficiency for AIs in the DareFightingICE using gRPC,” *arXiv Networking and Internet Architecture*, pp. 1- 5, 2023, doi : 10.1109/ICBIR57571.2023.10147629.

[12] C. H. V. P. Krishna and R. R. Kumar, “Implimentation Of Weighted Round Robin Load Balancing Algorithem For Grpc,” vol. 9, no. 7, pp. 1–9, 2022. [Online]. Available at : https://tijer.org/papers/TIJER2207001.pdf.

[13] S. Saberian, “Evaluating the performance and usability of HTTP vs gRPC in communication between microservices Najem Hamo,” no. May, p. 56, 2023. [Online]. Available at: https://www.diva-portal.org/smash/get/diva2:1768795/FULLTEXT02.

[14] M. Bolanowski *et al.*, “Eficiency of REST and gRPC Realizing Communication Tasks in Microservice-Based Ecosystems,” *Front. Artif. Intell. Appl.*, vol. 355, pp. 97–108, 2022, doi: 10.3233/FAIA220242.

[15] S. Sterman, M. J. Nicholas, and E. Paulos, “Towards Creative Version Control,” *Proc. ACM Human-Computer Interact.*, vol. 6, no. 2 CSCW, pp. 1–25, 2022, doi: 10.1145/3555756.

[16] R. Ls, W. Forrest, R. L. S. Frisbie, P. Grete, and F. W. Glines, “An Inquiry Approach to Teaching Sustainable Software Development with Collaborative Version Control,” 2022. [Online]. Available at: https://escholarship.org/uc/item/6fv1s464.

[17] A. Hinderks, F. J. Domínguez Mayo, J. Thomaschewski, and M. J. Escalona, “Approaches to manage the user experience process in Agile software development: A systematic literature review,” *Inf. Softw. Technol.*, vol. 150, no. September 2021, p. 106957, 2022, doi: 10.1016/j.infsof.2022.106957.

[18] L. Gren and P. Ralph, “What Makes Effective Leadership in Agile Software Development Teams?,” *Proc. - Int. Conf. Softw. Eng.*, vol. 2022-May, pp. 2402–2414, 2022, doi: 10.1145/3510003.3510100.

[19] R. Ouriques, K. Wnuk, T. Gorschek, and R. B. Svensson, “The role of knowledge-based resources in Agile Software Development contexts,” *J. Syst. Softw.*, vol. 197, p. 111572, 2023, doi: 10.1016/j.jss.2022.111572.

[20] K. Bernsmed, D. S. Cruzes, M. G. Jaatun, and M. Iovan, “Adopting threat modelling in agile software development projects,” *J. Syst. Softw.*, vol. 183, pp. 1–21, 2022, doi: 10.1016/j.jss.2021.111090.