Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024

# Increasing Accessibility and Scalability of Student Services Using Microservice Architecture: A Case Study on Developing a Timetable Service

Nino Beraia
*Georgian technical university, Professor*
n.beraia@gtu.ge

Oleg Smoliakov
*Georgian technical university, 4-th year student*
smoliakov.oleg23@gtu.ge

Andrei Bykov
*Georgian technical university, 4-th year student*
bikov.andrei24@gtu.ge

**Abstract**

This paper describes the development of a service for tracking the university schedule, implemented using a microservice architecture with a chat-bot as the client interface. During the development process, the needs of students and professors, as well as publicly available university data, were considered, allowing for the selection of an optimal platform and the creation of an interactive chat-bot interface.

All stages of the project's development are listed, starting from defining the main service objectives and ending with configuration and testing. Information is also provided about the chosen technologies and tools used during the development process, such as the Python and Java programming languages, the BeautifulSoup and Requests libraries, the Cloud Native and Domain-Driven Design architectural approaches, as well as containerization with Docker and orchestration with Kubernetes.

The service is designed to help students stay informed about upcoming classes and events by providing information in an accessible and intuitive format. It serves as a convenient addition to traditional methods of information delivery.

**Keywords**: Microservice Architecture, Chat-Bot, Information System

## Introduction

In the context of the digital transformation of the educational process, universities integrate more technologies to enhance students' experience with educational information. An important goal is to ensure data accessibility and simplify interaction with educational services in formats that are convenient for students. University websites remain the central element of the information ecosystem, however, there is a growing demand for additional tools, that can provide new methods of interaction. The challenges of modern educational services are:

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024

1. limited access to information due to outdated or slow systems, which hinders quick access to essential data;
2. low scalability, which prevents handling a growing number of users during peak periods, such as the beginning of a semester or exam sessions;
3. lack of personalization, making it impossible to adapt resources to individual student needs, which directly affects learning effectiveness;
4. low integration with other systems, creating a need for data duplication and significantly complicating the management of the educational process;
5. limited functionality in mobile versions, causing inconvenience for students who frequently use smartphones;
6. dependence on third-party development, which delays timely updates and adaptation to the university's specific needs.

The listed issues also affect commercial development, where accessible solutions have been found to enhance enterprise efficiency. The education sector can adopt these solutions for its own services. One key approach in this area is the creation of modular systems with a microservice architecture, composed of several independent but interconnected components. This structure enables efficient development and integration of systems with each other. The main advantages of this approach include:

1. division of responsibilities by decomposing the system into individual microservice components, simplifying work on each one by reducing the scope of requirements;
2. scalability, enabling easy identification of the most heavily loaded nodes within the system. Typically, the technology supports deploying multiple instances to distribute the load. Such horizontal scaling is most effective when the system is properly divided, with each service instance consuming minimal resources;
3. flexibility, provided by microservices offering maximum integration both with each other and with other systems, which enhances the ability to add new features to the overall system. This approach also allows for the connection of client interfaces of any format, whether they be web, desktop, or mobile applications, or even chat-bots in messaging platforms.

This paper examines the creation of a microservice project for organizing class schedules at the Georgian Technical University and delivering them through a chat-bot based on the Telegram messenger. The choice of schedule management as the service topic was driven by high user demand for this functionality within the current university system, as well as the general need to stay updated on relevant events, which is one of the most essential functions of any educational system. The choice of Telegram was based on surveys of international students, for whom this platform is more familiar and frequently used.

The advantages of the chosen architecture

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024

address the issues of similar systems: individual components can be horizontally scaled, which speeds up the system in the most heavily loaded areas; a focus on mobility is achieved without limiting the system to mobile platforms; and new features can be added by incorporating additional components. This makes it possible to create a chat-bot for another social network (e.g., Facebook), a mobile application, or a web interface, without requiring changes to the already developed services.

The following stages were implemented in the development of the service:

- defining the goals and objectives of the service;
- analysis of the audience and platform selection through studying the needs of students and professors to determine the most convenient and effective way to deliver information;
- development of service components: defining the structure and logic, integrating the service with the university system to obtain up-to-date information on class schedules and updates;
- development of the user interface: creating dialogue scenarios and defining the communication flow with the user;
- configuration and testing of the service: creating and configuring the service to meet all tasks and functions, conducting testing for compliance with requirements, and correcting errors.

At the same time, flexibility is required in building the system to enhance its reliability and potential for development. Modern software development standards allow for the selection of available technologies that can achieve the set goals:

- Python and Java were chosen as programming languages, both proven in microservice applications and ranked in the top three on the TIOBE index as of October 2024 [1];
- for efficient development in a microservice environment, the libraries BeautifulSoup (for parsing), Requests and Spring Web (for HTTP), and Aiogram (for asynchronous interaction with Telegram) were selected;
- modern development standards were applied, including Cloud Native, Domain-Driven Design, API-First, and asynchronous service interaction technologies;
- Kubernetes, a modern and proven tool, was chosen for application orchestration;
- The project used Docker containerization, a widely adopted technology in modern development;
- For continuous integration and deployment (CI/CD), GitHub Actions was used, currently managing nearly 78% of the source code market [2]. ArgoCD, a growing tool for CD, was also utilized.

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024

**Development of Timetable Service**

The primary functions that the service should perform were initially defined during the project design phase:

- Viewing the class schedule for a specific day or week.
- Notifications about upcoming classes and schedule changes.
- Search by subject or professor.
- Answers to frequently asked questions about the schedule, professors, and classrooms.
- Ability to contact an administrator in case of issues or questions.

These functions cover the primary needs of students, as noted in the questions and requests from 2nd-4th year international technical groups in Telegram chats.

To implement the microservice architecture, the service was divided into three components, each with specific responsibilities:

- Parser: an independent module responsible for scraping the university website using web scraping technology. This stateless service (does not store information) runs a limited number of times per day, stopping after successfully sending data, which minimizes load on the overall system
- Data Service: a microservice that processes, stores, and manages academic data, such as schedules, information about professors, and courses. It is the only stateful service, running continuously to manage database access and data, and providing an interface with the necessary contracts for other components.
- Chat-bot: serves as the client interface, enabling convenient student interaction with the system via the popular messenger, Telegram. The bot retrieves data from the main service and presents it to users in a user-friendly format. It also operates continuously and is stateless.

Service interaction is managed via HTTP using REST, allowing independent operation in Docker containers. Kubernetes provides orchestration, acting as a cluster for these containers.

The path from source code to the Kubernetes cluster is managed through GitHub CI. Configured GitHub Actions build a Docker image upon code push, which ArgoCD deploys as a Docker container in the Kubernetes cluster, ensuring zero downtime – keeping the old version active until the new one is fully operational. Multiple service copies can be launched based on configuration, distributing the load.

Integration with the university website is managed via the Parser service, which connects to the university server to check for data updates. The Timetable Service generates a new schedule and saves the data in a PostgreSQL database, linking it with existing static data (professors, rooms, study programs). The generated schedule is marked as the latest version. The Telegram Bot service retrieves data only when receiving a message in Telegram, by sending requests to the Timetable Service.

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024

Users can configure their preferred group, ensuring that only relevant group data is requested.

Figure 1 illustrates the development details and operation of the automated system for loading and processing data from the university's schedule page.

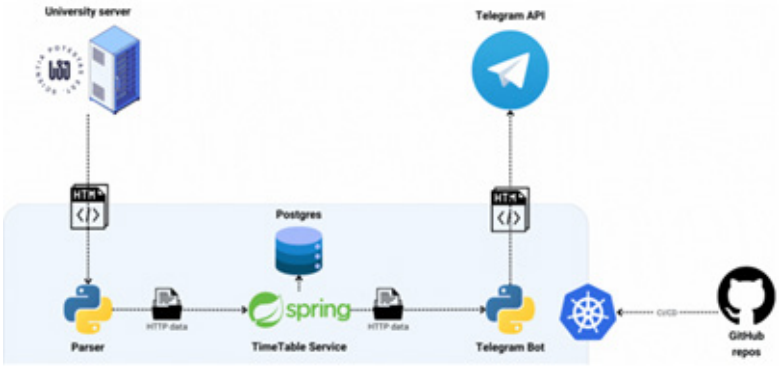


**Fig. 1.** Data Loading and Processing Diagram

As mentioned earlier, Argo CD was used for application deployment in Kubernetes. This tool makes it easy to deliver applications to the cluster by scanning for changes in the GitHub repository. When a new commit is detected, Argo generates new artifacts based on Kubernetes manifests.

Services and components are conveniently separated into Argo CD "Applications," each with customizable update rules and its own structure. Our setup includes several such applications (see Fig. 2):

- commons: contains access keys (database login and password, GitHub Packages access key);
- postgres: contains Kubernetes objects for running the Postgres database;
- parser, timetable, tgbot: contain Kubernetes objects for running the corresponding services.

Running a single microservice in Kubernetes may require multiple Kubernetes objects. To create these objects, Kubernetes manifests are written. Argo CD provides a clear view of the hierarchy of the created objects. Real examples are shown in Figures 3-5.

For the bot, manifests were written for a Secret with the Telegram token, a Deployment with deployment data, a Service for network access to the bot Pod, a ReplicaSet to manage replica count, and finally, a Pod – the container with the Python-based application.

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024

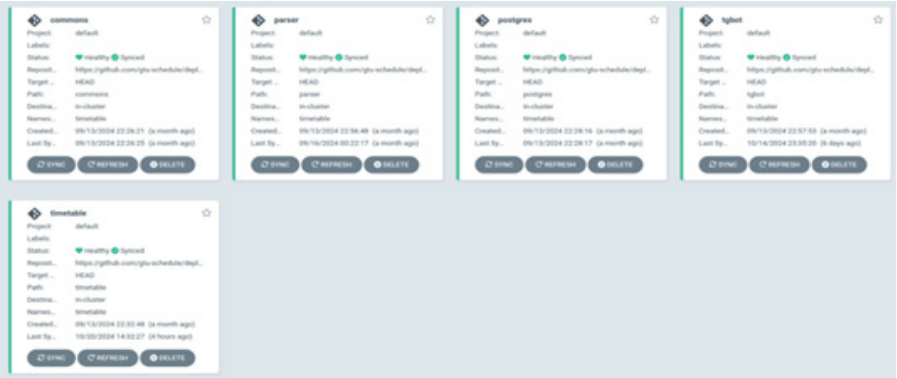


**Fig. 2**. Argo CD applications in Kubernetes cluster

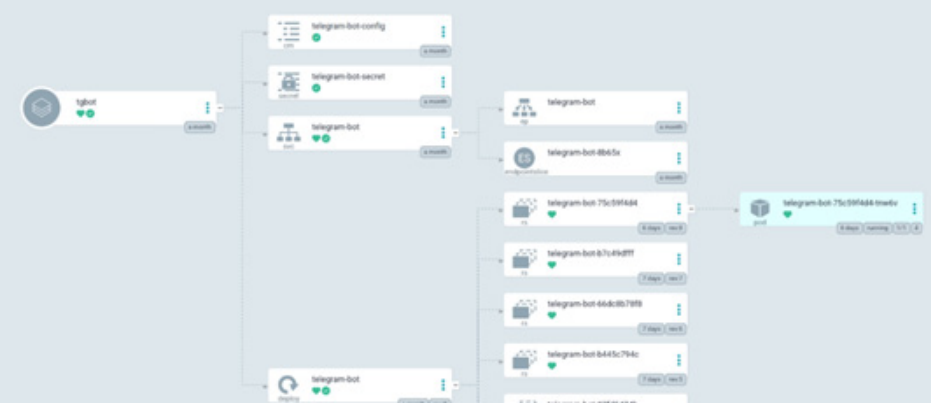


**Fig. 3**. Telegram Bot in Kubernetes cluster

The data parsing service has an interesting delivery model to the cluster. Since it doesn't need to run continuously, a scheduled task, CronJob, was set up. Periodically, a Job object is created, which triggers a Pod running the Python-based parser service. Once the task is completed, the Pod finishes its execution.

The chatbot's user interface was built using FSM (Finite State Machine). FSM-based interaction provides an efficient way to manage different application states and functions. In this setup, the bot's interface is designed so that users can easily complete the registration process and set up their schedule, after which they gain immediate access to both the weekly and the current day's schedules.


**Fig. 4**. Timetable service in Kubernetes cluster

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024


**Fig. 5.** Parser service in Kubernetes cluster

Registration starts with the “start” command, which opens a registration menu with three main buttons (see Fig. 6):

- **«**Language**»** – allows choosing one of the three languages used for instruction at the university.
- **«**Help**»** – displays a text with a description of all available bot commands;
- **«**Schedule setup**»** – The setup process includes several steps: selecting the language of instruction, academic degree, specialty, and group. Once completed, the settings are saved, and the menu is hidden. After registration, the bot enters the “waiting state” (see Fig. 7), where the user can choose between “Week schedule” or “Today’s schedule.” At any time, the user can re-register by sending the “/start” command.

There are also additional commands:

- **«language»** reopens the language selection menu;
- **«help»** displays a text guide with additional commands;
- **«about»** shows information about the bot and the schedule, as well as contact details for user support.

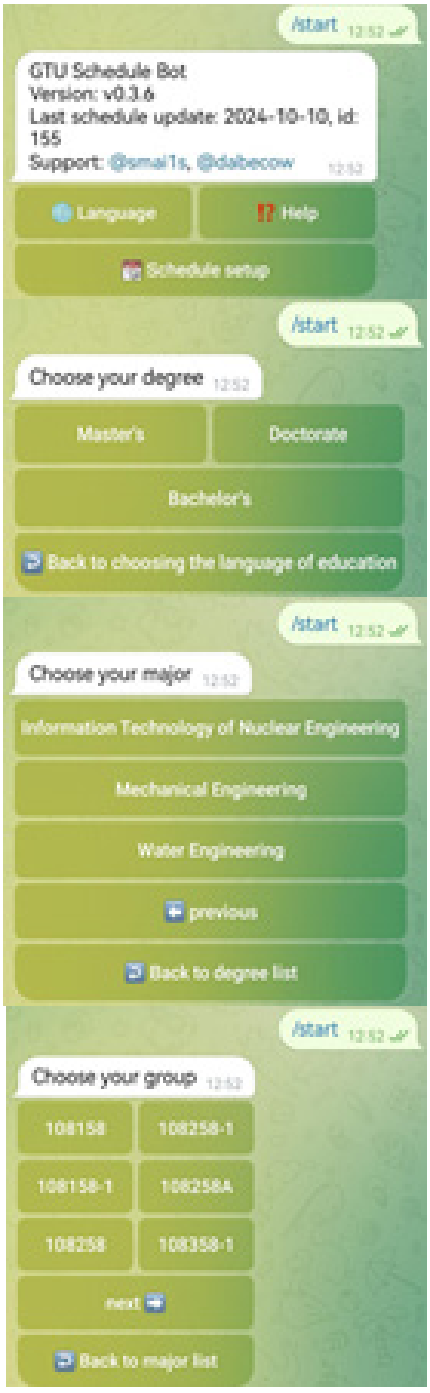


**Fig. 6.** Registration process

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024

This interface organization allows users to intuitively interact with the bot, following clearly defined steps, while FSM ensures easy control over states and transitions between them. An example of the generated schedule is shown in Fig. 7. The parsing and Telegram bot services are integrated

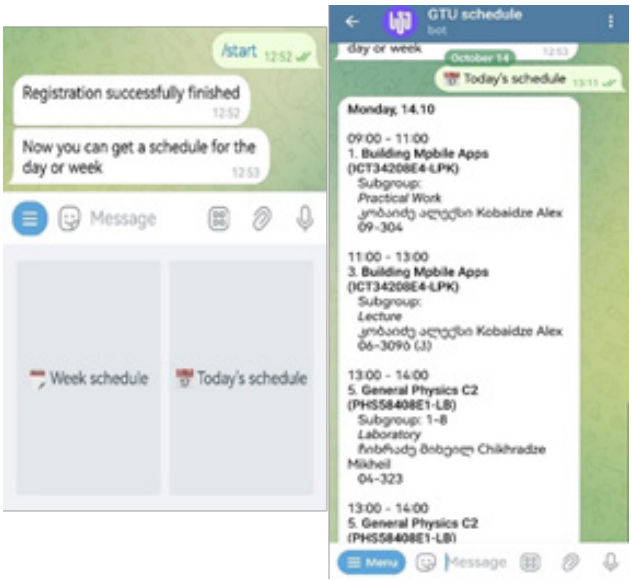


**Fig. 7.** The “waiting state” and an example of the schedule for the day are displayed

with the data service using the HTTP protocol and RestAPI architecture. The API-first approach allowed the interaction specification to be written first using OpenAPI, followed by parallel development of different services based on the described interface. Using a formalized specification makes development and debugging easier. For example, the Swag¬ger graphical interface was used to view and execute the API of the service generating or using the OpenAPI specification.

For example, in Figure 8, the requests for retrieving and creating courses, groups, and classes are shown. The creation requests are actively used by the parser service, which adds data to the data service, while the data retrieval requests are generated by the chatbot service, which needs to display the data to users.

Similarly, the bot provides an API for the data service, allowing it to send notifications to users, such as informing them about changes in the schedule.

Thus, microservices written in different languages freely exchange information.

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024

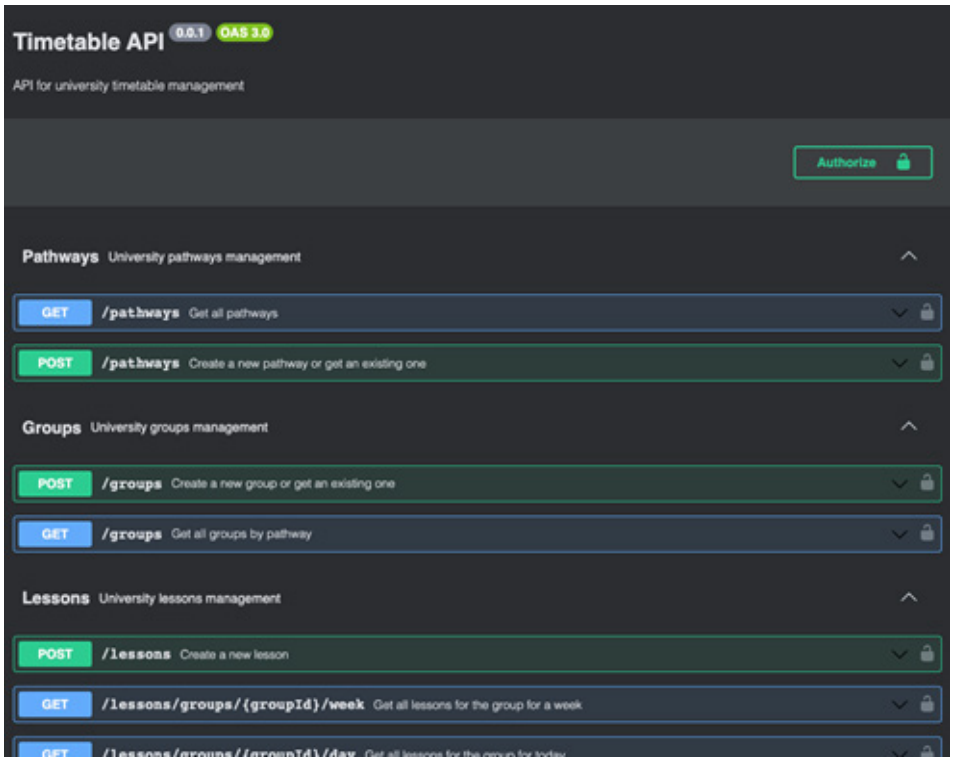


**Fig. 8.** Graphical representation of Data Service API

## Conclusion

The use of a microservices approach in higher education services opens up great opportunities to improve the quality and accessibility of education. It enhances the quality of systems by increasing flexibility and modularity, speeds up development in student teams, and ensures greater reliability compared to traditional approaches.

The created system will undoubtedly be of interest to those who value innovation, modern technologies, and aim to improve the educational process. The experience of such development will be valuable for future educational system projects in student teams, serving as an example of independent development of modular systems.

In conclusion, the established process allowed for quick deployment of changes to the server, with new features being added independently to the modules. This ensured that access to the system was not restricted, and users could continue using the bot at any time.

At the end of the month, during which the bot's information was shared within a small circle of students, the total number of users reached 60, with 32 using it in the last week at the time of this conclusion. At least half of the users interacted with the bot at least once a week throughout its operation, indicating strong interest in the service, even without a mass information distribution

In the future, there are plans to develop

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 2, 2024

other client parts, such as a web client and chatbots for other social networks like Facebook, to maximize audience reach. The functionality of the system will also be expanded, including features like the ability to sign up for consultations with professors, create custom schedules, and more. To achieve this, the involvement of more interested students is planned, who will continue to develop the university system.

The implementation of such technologies will undoubtedly help create a much more attractive educational environment, fostering the development of future generations. It will assist students in receiving a higher quality education, better handling academic tasks, and gaining more practical experience, which will positively impact their academic performance and motivation to learn.

**References**

N. Beraia, L. Tokadze, I. Aptsiauri, M. Shiukashvili. Prospects and problems of using artificial intelligence in higher education. 10th International new york conference on evolving trends in interdisciplinary research & practices. Manhattan, New York City, IKSAD Publications - 2024©, Issued: 24.06.2024, Proceeding book, 308-313 pp. ISBN: 978-625-367-739-8

Beraia N., Shekunov A., Timofeev P., Tokadze L. On the issue of effective use of information technologies in the adaptation of foreign students and the optimization of educational processes. Scientific discussion (Praha, Czech Republic), Vol 1, No 82, (2023), p. 32-36. ISSN 3041-4245; DOI: 10.5281/zenodo.10117504

TIOBE Index for October 2024 (accessed 21.10.2024). https://www.tiobe.com/tiobe-index/

Market Share of Github (accessed 21.10.2024). https://6sense.com/tech/source-code-management/github-market-share#