UDC 004.75
DOI https://doi.org/10.32782/2521-6643-2025-2-70.17

**Oleksiichuk Yu. F.,** Candidate of Physical and Mathematical Sciences, Associate Professor at the Department of Computer Science and Information Technology
Poltava University of Economics and Trade
ORCID: 0000-0002-0585-3307

# DESIGN AND IMPLEMENTATION OF A CHATBOT FOR PROSPECTIVE STUDENTS

*This paper addresses the pressing issue of digitalizing communication processes in higher education, focusing on the development and implementation of an intelligent chat-bot designed to facilitate interaction with prospective students of the Computer Science educational program. In the context of increasing competition among higher education institutions and the growing need to provide year-round informational support for applicants, the study substantiates the feasibility of employing the Telegram messenger as a platform for automated consultation.*

*The paper analyzes the key shortcomings of traditional communication channels with potential applicants, including their temporal and geographical constraints, fragmentation of information across different web resources, and the excessive workload imposed on admission office staff and academic personnel.*

*The software solution was developed using the Java programming language and the Spring Boot framework. The system is designed according to a multilayered architecture that follows the Controller–Service–Repository design pattern, ensuring modularity, scalability, and ease of maintenance. Integration with the Telegram API is achieved through the Telegram Bots library, while data persistence is managed via the Oracle Database management system.*

*The chat-bot's functional capabilities include providing information about the advantages of the educational program and the university, specific admission requirements for school and college graduates, features of studying at different academic levels, current tuition fees, scholarship and grant opportunities, and the option to subscribe to important updates. The request-handling mechanism, based on user session management and dedicated message handlers, enables a multi-stage interaction flow with logical transitions between dialogue states.*

*For administrative management, the system incorporates an interface built with the Swagger library, which provides automatically generated interactive API documentation and supports the execution of test requests directly through a web interface.*

*The results of the research confirm the effectiveness of the proposed approach in addressing current challenges related to the digitalization of higher education. The developed system significantly reduces the workload of administrative personnel, ensures applicants' round-the-clock access to essential information, and improves the overall quality of communication between the university and prospective students.*

Key words: *chatbot, telegram, java, spring.*

***Олексійчук Ю. Ф. Розробка та впровадження чат-бота для майбутніх студентів***

*Стаття присвячена актуальній проблемі цифровізації комунікаційних процесів у сфері вищої освіти, зокрема питанням розробки та впровадження інтелектуального чат-бота для взаємодії з майбутніми студентами освітньої програми з комп'ютерних наук. У контексті зростаючої конкуренції між закладами вищої освіти та необхідності забезпечення цілорічної інформаційної підтримки абітурієнтів обґрунтовується доцільність використання месенджера Telegram як платформи для автоматизованого консультування.*

*Проаналізовано основні недоліки традиційних каналів комунікації з потенційними вступниками, зокрема їхню часову та географічну обмеженість, фрагментованість інформації на різних веб-ресурсах, а також значне навантаження на співробітників приймальних комісій та викладачів.*

*Розроблено програмне рішення, яке реалізоване з використанням мови програмування Java та фреймворку Spring Boot. Система побудована за багатошаровою архітектурою з застосуванням патерну проектування Controller–Service–Repository, що забезпечує модульність, масштабованість та зручність супроводу програмного продукту. Для інтеграції з Telegram API використано бібліотеку Telegram Bots, а для зберігання даних – систему управління базами даних Oracle Database.*

*Функціональні можливості чат-бота включають надання інформації про переваги освітньої програми та університету, особливості вступу після школи та коледжу, характеристики навчання на різних освітніх рівнях, актуальну вартість навчання, стипендіальні та грантові програми, а також можливість підписки на важливі повідомлення.*

*Механізм обробки користувацьких запитів через систему обробників та управління сесіями користувачів дозволяє реалізувати багатоетапну взаємодію з логічними переходами між різними станами діалогу.*

*Для адміністративного управління системою впроваджено інтерфейс на основі бібліотеки Swagger, який надає автоматично згенеровану інтерактивну документацію API та можливість виконання тестових запитів безпосередньо через веб-інтерфейс.*

*Результати дослідження підтверджують ефективність запропонованого підходу для вирішення актуальних викликів цифровізації вищої освіти. Розроблена система дозволяє суттєво знизити навантаження на адміністративний персонал, забезпечити абітурієнтам цілодобовий доступ до необхідної інформації та покращити загальну якість комунікації між університетом та майбутніми студентами.*
Ключові слова: *чат-бот, telegram, java, spring.*

**Formulation of the problem.** In the contemporary context of digitalization in higher education, universities are increasingly compelled to transform traditional approaches to communication with prospective applicants. Interaction with future students constitutes a continuous process that extends beyond the admission campaign period and encompasses the full cycle of career guidance activities throughout the year. Particularly in highly competitive fields such as computer science, cultivating a positive perception of the educational program and institution is essential for attracting motivated and talented applicants.

Conventional communication channels, including telephone calls to admissions offices, email correspondence, and participation in open house events, are constrained by temporal and geographical limitations. Prospective students, however, expect timely responses to inquiries concerning admission requirements, program characteristics, the advantages of bachelor's or master's degree studies, tuition fees, and scholarship opportunities at any time of the year, irrespective of office hours or geographical location.

Staff members of admissions offices and academic departments face a considerable informational burden, as they are required to provide repetitive responses to routine inquiries about the computer science program throughout the academic year. This situation diminishes the quality of services, increases response times, and results in inefficient use of human resources that could otherwise be allocated to addressing more complex and non-standard issues.

Additionally, information regarding admission is often fragmented across various web pages and formats, necessitating significant effort for searching and systematization. Applicants, particularly recent school and college graduates, may encounter difficulties navigating multiple information sources, which negatively influences their interaction experience with the university even prior to the admission stage.

The significance of this issue is further heightened by the growing competition among higher education institutions for talented students. Universities that succeed in providing a high level of service and convenient communication with prospective applicants secure a competitive advantage in the educational market. Consequently, the implementation of innovative digital interaction tools is becoming indispensable for maintaining institutional competitiveness and fostering a positive institutional image.

In the contemporary digital environment, chatbots [1] are becoming an increasingly widespread tool of communication among adolescents and students. In particular, a report by *Common Sense Media* (USA, spring–summer 2025), based on a representative sample of more than 1,000 adolescents aged 13–17, indicates that 72% of American teenagers have experimented with AI companions (e.g., Character.AI, Replika, ChatGPT), while 52% have become regular users. Among those who use chatbots regularly, 13% engage with them daily, and 21% interact with them several times per week [2].

**Analysis of recent research and publications.** Contemporary Ukrainian youth actively employ a variety of channels for accessing news and digital content, including Telegram (with chatbots), YouTube, Instagram, Viber, and traditional television. Despite the documented security risks associated with the use of Telegram during wartime [3], as long as this messenger remains unrestricted, its popularity and potential applications across diverse domains must be considered. These include finance [4], education [5–9], the tourism industry [9], complaint management [10], system monitoring [11–12], and healthcare [13], among others. Telegram bots provide a user-friendly interface, and their underlying logic can be implemented in a wide range of programming languages [14]. Owing to this simplicity, Telegram is frequently utilized as an interface for the Internet of Things [12, 15–16].

**The purpose of this article** is the design and implementation of a Telegram bot for prospective applicants to the Computer Science program. The central idea is to provide only the most essential information in a concise and accessible form. The key functionalities offered to applicants include:

- obtaining information about the advantages of the educational program and the university;
- learning about the specifics of admission after college and after high school;
- exploring the features of bachelor's and master's studies;
- requesting additional information and submitting questions;
- accessing up-to-date information on tuition fees, scholarship and grant opportunities, as well as possibilities for free education;
- subscribing to important announcements and notifications.

**Presenting main material.** The development of the chatbot was carried out using the Java programming language in combination with the Spring Boot framework. Database operations are managed through Spring Data JPA, while security and access control are ensured by Spring Security. The processing of API requests is implemented using Spring Web, thus forming a robust and modular backend architecture.

To facilitate integration with the Telegram API, the Telegram Bots library is employed [17]. This library provides a lightweight yet effective Java framework for seamless interaction with the Telegram Bots API, thereby

streamlining the development of intelligent software agents. Its functionality encompasses built-in mechanisms for token management, the use of the Jetty HTTP client for reliable communication, and extensive documentation that supports efficient implementation and maintenance.

To store data, the Oracle Database is employed [18-19]. The database is used exclusively for preserving information about users who have expressed interest in the specialization. The Entity–Relationship (ER) diagram of the database is presented in Figure 1.

The Tg_Users table contains information about users obtained via the Telegram API [20]. The key attribute is userId, which is used for user identification. In addition, the Telegram API provides access to the user's nickname, first and last name as registered in Telegram, and information indicating whether the account belongs to a bot. However, this information is often incomplete or hidden by users.

The Connections table stores records of all interactions with the Telegram bot. These data are applied to analyze the popularity of the software product.

The Lead table contains information about user activities that require further processing by the department staff. This includes requests for assistance or specific questions that the chatbot could not resolve.

The Subscriber table stores information about Telegram subscribers. Subscribers receive important updates regarding university admission directly through Telegram.

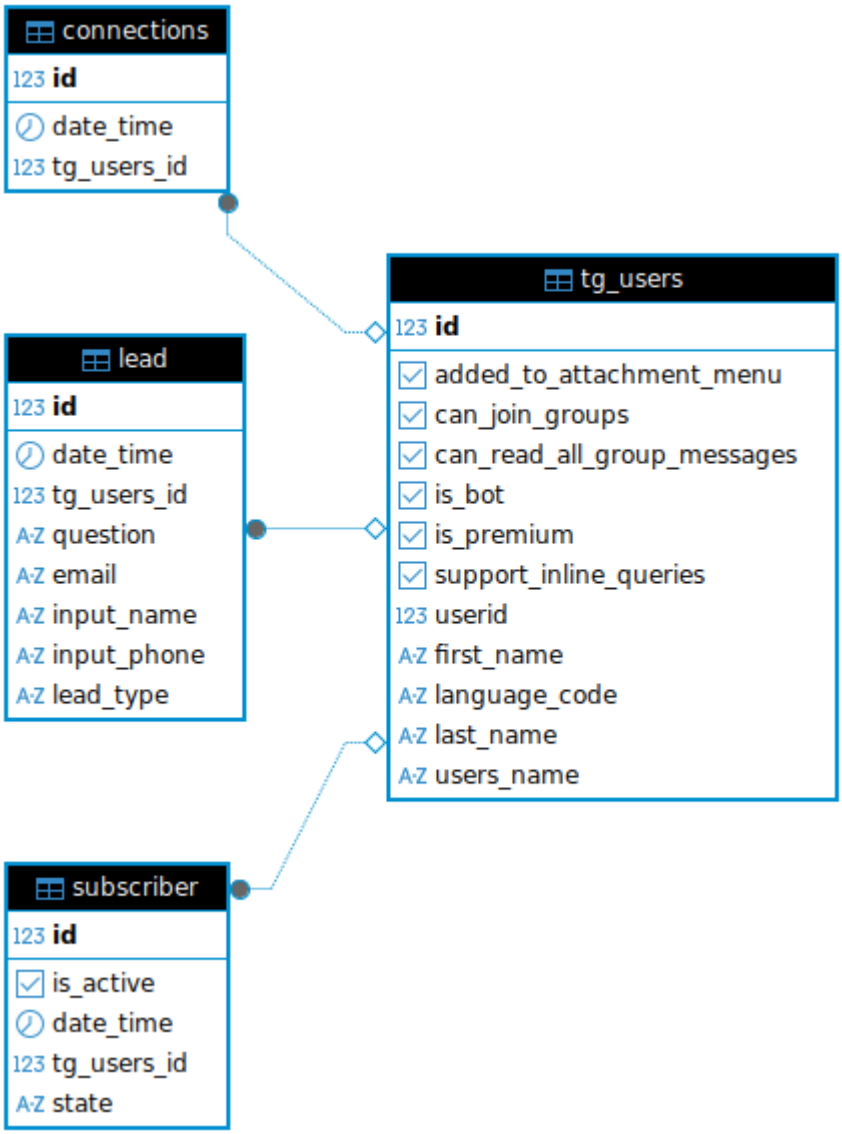


Fig. 1. ER diagram of the database

The chatbot logic is implemented within the bot package. The TelegramBot class provides overall control of the chatbot and extends the TelegramLongPollingBot class from the Telegram Bots library [17]. Specifically, this class overrides the onUpdateReceived(Update update) method, which is responsible for processing user messages.

Upon receiving the initial message from a user, an object of the UserSession class is created, which stores information regarding the current stage of the user's interaction.

```
@Override
public void onUpdateReceived(Update update) {
    Long chatId;
    User user;
    if(update.hasMessage() && update.getMessage().hasText()) {
        user = update.getMessage().getFrom();
        chatId = update.getMessage().getChatId();
    } else  if(update.hasCallbackQuery()
        && update.getCallbackQuery().getData()!=null){
        user = update.getCallbackQuery().getFrom();
        chatId = update.getCallbackQuery().getMessage().getChatId();
    }
    else {
        log.warn("Unexpected update from user. Not message or not text");
        return;
    }
    UserSession session = userSessionService.getSession(chatId);
    userSessionService.saveSession(chatId,session);
    UserRequest userRequest = UserRequest
            .builder()
            .update(update)
            .userSession(session)
            .chatId(chatId)
            .user(user)
            .build();
    boolean dispatched = dispatcher.dispatch(userRequest);
    if (!dispatched) {
        if (session.getState().isAllowText()){
            textRequestHandler.handle(userRequest);
        }else {
            log.warn("Unexpected update from user");
        }
    }
}
```

At each stage of interaction, the user receives specific textual information along with a set of buttons that facilitate further navigation. The set of buttons is stage-dependent and is generated through the KeyboardHelper class. In certain stages, users are also permitted to provide textual input, for example, when submitting a question or entering their name.

Each stage is associated with a dedicated Handler, responsible for processing user messages, which extends the abstract class UserRequestHandler.

```
public abstract class UserRequestHandler {
    public abstract boolean isApplicable(UserRequest request);
    public abstract void handle(UserRequest dispatchRequest);
    public abstract boolean isGlobal();
    public boolean isCommand(Update update, String command) {
        return (update.hasMessage() && update.getMessage().isCommand()
                && update.getMessage().getText().equals(command))
                || (update.hasCallbackQuery() && update.getCallbackQuery().getData()!=null
                && update.getCallbackQuery().getData().equals(command));
    }
}
```

Within a Handler, the content delivered to the user at the current stage is defined, along with the permissible set of commands. Each command typically triggers a transition to another stage of the interaction.

Structurally, the code of the web service follows a multilayered architecture, with each layer responsible for a specific functionality. The application implements the Controller–Service–Repository design pattern, which is widely adopted in modern enterprise software development to ensure modularity, maintainability, and scalability [21-23].

The Controller layer is responsible for handling incoming HTTP requests and mapping them to the corresponding application functionality. Within this layer, input data is validated and converted into domain-specific structures before being passed further for processing. Controllers serve as the main entry point to the system, enabling interaction between end users and the application logic.

The Service layer encapsulates the core business logic. It coordinates operations, enforces domain rules, and ensures the correct execution of workflows. Centralizing business functionality in services promotes separation of concerns, improves system clarity, and simplifies debugging, testing, and future extension.

The Repository layer manages data persistence and retrieval. It abstracts the database operations by providing a standardized interface for entity management. In this project, repositories rely on Spring Data JPA, which minimizes the amount of boilerplate code, automatically generates query implementations, and integrates seamlessly with relational databases.

The chosen framework, Spring Boot, plays a crucial role in supporting this architecture. It provides built-in mechanisms for dependency injection, configuration management, and seamless integration of various Spring modules, such as Spring Web, Spring Data JPA, and Spring Security. This ensures that the multilayered design is not only logically consistent but also technically efficient, with reduced development overhead and enhanced system reliability.

This layered approach increases modularity and fosters reusability of components, while reducing coupling between the different parts of the system. As a result, the overall architecture achieves higher robustness, maintainability, and adaptability to future requirements.

For the purpose of informing administrators, the system implements functionality for automatically sending both statistical reports and user-submitted messages that require attention to designated Telegram addresses. This ensures that administrators receive timely notifications without the need for manual monitoring of the system.

The delivery of this information is automated through the use of the scheduling mechanism provided by the Spring Framework, specifically the @Scheduled annotation available in Spring Boot [24]. This annotation allows developers to define tasks that are executed periodically or at specific times, following either fixed intervals, fixed delays, or cron expressions. As a result, the application can be configured to send summaries of user activity, error logs, or other relevant updates at predefined times, thereby facilitating efficient monitoring and decision-making.

One of the key advantages of using the Spring scheduling mechanism is its flexibility. Schedules can be easily adjusted through configuration parameters, enabling administrators to change the frequency of reporting without modifying the underlying source code. Moreover, multiple scheduled tasks can coexist within the application, allowing for the automation of diverse processes, such as data synchronization, report generation, and system health checks.

Another significant feature of the @Scheduled mechanism is its integration with Spring's dependency injection and transaction management. This ensures that scheduled tasks operate consistently within the broader application context, maintaining access to necessary services, repositories, and security mechanisms. Furthermore, in production environments, scheduling can be combined with asynchronous execution and thread pool management, thereby improving system scalability and preventing long-running tasks from blocking critical operations.

Overall, the use of scheduled tasks in Spring Boot provides a reliable and extensible mechanism for automating routine processes. In the context of this project, it enables administrators to remain informed about user activity and system status, while simultaneously reducing the workload associated with manual monitoring and repetitive tasks. This contributes to higher system efficiency, improved responsiveness to user needs, and enhanced overall quality of service.

An example of the practical implementation of scheduled tasks in this project is illustrated in the code fragment below. The class LeadsScheduler is annotated with @Component, which makes it a managed Spring Bean and enables its automatic detection during application context initialization. The @Scheduled annotation is applied to the execute() method, specifying a cron expression that determines the exact schedule of task execution. In this case, the method is configured to run daily at 14:10.

```
@Component
@RequiredArgsConstructor
@Slf4j
public class LeadsScheduler {
    private final LeadsScheduledService leadsScheduledService;
    @Scheduled(cron = "0 10 14 * * ?")
   public void execute() {
        try {
             leadsScheduledService.sendAdminDailyInfo();
        }
        catch (Exception e){
            log.error(e.getMessage());
        }
    }
}
```

The method delegates the execution logic to the service layer (LeadsScheduledService), which encapsulates the functionality of preparing and sending daily information to administrators. This approach follows the principles of layered architecture by separating scheduling responsibilities from business logic. For fault tolerance, exception handling is incorporated, and errors are logged using the SLF4J logging framework [25]. Such an implementation ensures both reliability and maintainability of the scheduling mechanism.

For administrative purposes, a dedicated management interface was implemented using the Swagger library [26]. This tool was selected because it provides the capability to rapidly establish a functional administrative environment with minimal configuration effort: integration into the project requires only the inclusion of the library and a small amount of additional code.

Beyond its ease of integration, Swagger offers significant advantages for system maintenance and oversight. It provides automatically generated and interactive API documentation, allowing administrators not only to review available endpoints but also to execute test requests directly through the interface. This functionality simplifies validation, debugging, and monitoring of the system's behavior in real time.

Although its interface is relatively straightforward, Swagger ensures that administrators can perform all essential management operations effectively.

**Conclusions.** The conducted research demonstrates the effectiveness of using digital tools, in particular Telegram chat-bots, for improving communication between higher education institutions and prospective students. The developed system provides applicants with quick access to essential information about admission requirements, study opportunities, tuition fees, and scholarship programs, thereby reducing the workload on administrative staff and enhancing the quality of interaction with the university.

The implemented architecture, based on Java and Spring Boot, ensures a modular and extensible structure of the software solution. Integration with the Telegram Bots API and the use of supporting technologies such as Spring Data JPA, Spring Security, and scheduled task execution mechanisms allow for reliable and efficient functionality. The inclusion of an administrative interface based on Swagger additionally increases system transparency and facilitates management processes.

The results confirm that the proposed approach is an effective means of addressing current challenges in the digitalization of higher education. Future work will focus on expanding system capabilities, including the integration of AI-based systems into the chat-bot, which would enable more intelligent and personalized interaction with users. Such improvements are expected to further increase the attractiveness of the educational program and strengthen the competitive position of the university.

**Bibliography:**


1. Adamopoulou E., Moussiades L. An overview of chatbot technology. In *IFIP international conference on artificial intelligence applications and innovations* (pp. 373–383). Cham: Springer International Publishing. 2020. https://doi.org/10.1007/978-3-030-49186-4_31

2. Perez S. 72% of U.S. teens have used AI companions, study finds. TechCrunch. 2025. URL: https://techcrunch.com/2025/07/21/72-of-u-s-teens-have-used-ai-companions-study-finds

3. Balovsyak N. Anonymous and official Telegram channels in Ukraine: analysis of popularity during the hybrid war. Current Issues of Mass Communication, 37, 2025. P. 30–42. https://doi.org/10.17721/CIMC.2025.37.30-42

4. Haitan O. M., SnytkaI. V. Integrated platforms for automating personal financial accounting based on chatbots and cloud technologies. *Systems and Technologies*, 2025, 69(1). P. 58–70. https://doi.org/10.32782/2521-6643-2025-1-69.7

5. Olhovska O. V., Chernenko O. O., Ananenko I. V., Parfonova T. O., Rudenko N. S. Development of a training simulator for system analysis in the form of a chat-bot. *Visnyk of Kherson National Technical University*, (2), 2023. P. 196–202. https://doi.org/10.35546/kntu2078-4481.2023.2.27

6. Ismawati D., Prasetyo I. The development of Telegram bot through short story. In *Brawijaya International Conference on Multidisciplinary Sciences and Technology (BICMST 2020)* (pp. 209–212). Atlantis Press. https://doi.org/10.2991/assehr.k.201021.049

7. Rianto R., Rahmatulloh A., Firmansah T. A. Telegram Bot Implementation in Academic Information Services with The Forward Chaining Method. *Sinkron : Jurnal Dan Penelitian Teknik Informatika*, *3*(2), 2019. P. 73–78. https://doi.org/10.33395/sinkron.v3i2.10023

8. Aisyah R. N., Istiqomah D. M., Muchlisin M. Developing e-learning module by using telegram bot on ICT for ELT course. In *5th International Conference on Arts Language and Culture (ICALC 2020)* (pp. 106–111). Atlantis Press. https://doi.org/10.2991/assehr.k.210226.054

9. Olkhovska O. V., Oleksiichuk Y. F., Koshova O. P., Chernenko O. O., Boiko O. A. Development of a telegram chat-bot for providing technical support in the field of tourist services. *Taurida Scientific Herald. Series: Technical Sciences*, (6), 2024. P. 35–44. https://doi.org/10.32782/tnv-tech.2023.6.5

10. Rosid M. A., Rachmadany A., Multazam M. T., Nandiyanto A. B. D., Abdullah A. G., Widiaty, I. Integration telegram bot on e-complaint applications in college. In *IOP conference series: Materials Science and Engineering* (Vol. 288, No. 1, p. 012159). IOP Publishing. 2018. https://doi.org/10.1088/1757-899X/288/1/012159

11. Idhom M., Fauzi A., Alit R., Wahanani H. E. Implementation system telegram bot for monitoring Linux server. In *International conference on science and technology (ICST 2018)* (pp. 1089–1093). Atlantis Press. https://doi.org/10.2991/icst-18.2018.219

12. Bestari D. N., Wibowo A. An IoT-Based Real-Time Weather Monitoring System Using Telegram Bot and Thingsboard Platform. *International Journal of Interactive Mobile Technologies*, *17*(6). 2023. https://doi.org/10.3991/ijim.v17i06.34129

13. Djoelianto A. D., Kautsar I. A., Rosid M. A. Development of Web Service and Telegram Bot for Location-Based Health Service Information System. *Procedia of Engineering and Life Science*, *2*(2). 2022. https://doi.org/10.21070/pels.v2i2.1280

14. Modrzyk N. *Building telegram bots: develop bots in 12 programming languages using the telegram bot API*. Apress. 2018. https://doi.org/10.1007/978-1-4842-4197-4

15. De Oliveira J. C., Santos D. H., Neto M. P. Chatting with Arduino platform through telegram bot. In *2016 IEEE International Symposium on Consumer Electronics (ISCE)* (pp. 131–132). IEEE. https://doi.org/10.1109/ISCE.2016.7797406

16. Zaid, M. I. M. A., Abdullah, R., Ismail, S. I., Dzulkefli, N. N. S. N. IoT-based emergency alert system integrated with telegram bot. In *2023 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)* (pp. 126-131). IEEE. https://doi.org/10.1109/I2CACIS57635.2023.10193550

17. Bermudez, R. *TelegramBots (v.6.7.0)* [Computer software]. 2023. GitHub. URL: https://github.com/rubenlagus/TelegramBots

18. Kothuri R., Godfrind A., Beinat E. Pro oracle spatial for oracle database 11g. Apress. 2007. URL: https://link.springer.com/book/9781430242871

19. Maksymchuk S., Kabak L., Moroz B. Using of the modern data mining technics in customs of Ukraine. *Systems and Technologies,* 2(58), 2019. P. 33–49. https://doi.org/10.32836/2521-6643-2019-2-58-2

20. Khaund T., Hussain M. N., Shaik M., Agarwal N. Telegram: Data collection, opportunities and challenges. In Annual international conference on information management and big data (pp. 513–526). Cham: Springer International Publishing. 2020. https://doi.org/10.1007/978-3-030-76228-5_37

21. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Longman Publishing Co., Inc., USA. 2002. URL: https://dl.acm.org/doi/abs/10.5555/579257

22. Oleksiichuk Y. F., Olkhovska O. V., Olkhovsky D. M., Orlova D. I. Design and development of a web service for generating and sending pdf documents. *Systems and Technologies*, *65*(1), 2023. P. 39–45. https://doi.org/10.32782/2521-6643-2023.1-65.5

23. Walls C. Spring in action. Simon and Schuster. 2022.

24. Krnac L. Tasks and Scheduling. In: Pivotal Certified Spring Enterprise Integration Specialist Exam. Apress, Berkeley, CA. 2015. https://doi.org/10.1007/978-1-4842-0793-2_1

25. Juneau J. Exceptions and Logging. In: Java 9 Recipes. Apress, Berkeley, CA. 2017. https://doi.org/10.1007/978-1-4842-1976-8_9

26. Dos Santos J. S., Azevedo L. G., Soares E. F., Thiago R. M., da Silva V. T. Analysis of Tools for REST Contract Specification in Swagger/OpenAPI. In ICEIS (2) (pp. 201–208). 2020. https://doi.org/10.5220/0009381202010208

**References:**

1. Adamopoulou, E., & Moussiades, L. (2020, May). An overview of chatbot technology. In *IFIP international conference on artificial intelligence applications and innovations* (pp. 373–383). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-49186-4_31

2. Perez, S. (2025, July 21). 72% of U.S. teens have used AI companions, study finds. TechCrunch. Retrieved from: https://techcrunch.com/2025/07/21/72-of-u-s-teens-have-used-ai-companions-study-finds

3. Balovsyak, N. (2025). Anonymous and official Telegram channels in Ukraine: analysis of popularity during the hybrid war. *Current Issues of Mass Communication*, 37, 30–42. https://doi.org/10.17721/CIMC.2025.37.30-42

4. Haitan, O. M., & Snytka, I. V. (2025). Integrated platforms for automating personal financial accounting based on chatbots and cloud technologies. *Systems and Technologies*, *69*(1), 58–70. https://doi.org/10.32782/2521-6643-2025-1-69.7

5. Olhovska, O. V., Chernenko, O. O., Ananenko, I. V., Parfonova, T. O., & Rudenko, N. S. (2023). Development of a training simulator for system analysis in the form of a chat-bot. *Visnyk of Kherson National Technical University*, (2), 196–202. https://doi.org/10.35546/kntu2078-4481.2023.2.27

6. Ismawati, D., & Prasetyo, I. (2020, October). The development of Telegram bot through short story. In *Brawijaya International Conference on Multidisciplinary Sciences and Technology (BICMST 2020)* (pp. 209–212). Atlantis Press. https://doi.org/10.2991/assehr.k.201021.049

7. Rianto, R., Rahmatulloh, A., & Firmansah, T. A. (2019). Telegram Bot Implementation in Academic Information Services with The Forward Chaining Method. *Sinkron : Jurnal Dan Penelitian Teknik Informatika*, *3*(2), 73–78. https://doi.org/10.33395/sinkron.v3i2.10023

8. Aisyah, R. N., Istiqomah, D. M., & Muchlisin, M. (2021, March). Developing e-learning module by using telegram bot on ICT for ELT course. In *5th International Conference on Arts Language and Culture (ICALC 2020)* (pp. 106–111). Atlantis Press. https://doi.org/10.2991/assehr.k.210226.054

9. Olkhovska, O. V., Oleksiichuk, Y. F., Koshova, O. P., Chernenko, O. O., & Boiko, O. A. (2024). Development of a telegram chat-bot for providing technical support in the field of tourist services. *Taurida Scientific Herald. Series: Technical Sciences*, (6), 35–44. https://doi.org/10.32782/tnv-tech.2023.6.5

10. Rosid, M. A., Rachmadany, A., Multazam, M. T., Nandiyanto, A. B. D., Abdullah, A. G., & Widiaty, I. (2018). Integration telegram bot on e-complaint applications in college. In *IOP conference series: Materials Science and Engineering* (Vol. 288, No. 1, p. 012159). IOP Publishing. https://doi.org/10.1088/1757-899X/288/1/012159

11. Idhom, M., Fauzi, A., Alit, R., & Wahanani, H. E. (2018, December). Implementation system telegram bot for monitoring Linux server. In *International conference on science and technology (ICST 2018)* (pp. 1089–1093). Atlantis Press. https://doi.org/10.2991/icst-18.2018.219

12. Bestari, D. N., & Wibowo, A. (2023). An IoT-Based Real-Time Weather Monitoring System Using Telegram Bot and Thingsboard Platform. *International Journal of Interactive Mobile Technologies*, *17*(6). https://doi.org/10.3991/ijim.v17i06.34129

13. Djoelianto, A. D., Kautsar, I. A., & Rosid, M. A. (2022). Development of Web Service and Telegram Bot for Location-Based Health Service Information System. *Procedia of Engineering and Life Science*, *2*(2). https://doi.org/10.21070/pels.v2i2.1280

14. Modrzyk, N. (2018). *Building telegram bots: develop bots in 12 programming languages using the telegram bot API*. Apress. https://doi.org/10.1007/978-1-4842-4197-4

15. De Oliveira, J. C., Santos, D. H., & Neto, M. P. (2016, September). Chatting with Arduino platform through telegram bot. In *2016 IEEE International Symposium on Consumer Electronics (ISCE)* (pp. 131-132). IEEE. https://doi.org/10.1109/ISCE.2016.7797406

16. Zaid, M. I. M. A., Abdullah, R., Ismail, S. I., & Dzulkefli, N. N. S. N. (2023, June). IoT-based emergency alert system integrated with telegram bot. In *2023 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)* (pp. 126-131). IEEE. https://doi.org/10.1109/I2CACIS57635.2023.10193550

17. Bermudez, R. (2023). *TelegramBots (v.6.7.0)* [Computer software]. GitHub. https://github.com/rubenlagus/TelegramBots

18. Kothuri, R., Godfrind, A., & Beinat, E. (2007). Pro oracle spatial for oracle database 11g. Apress. Retrieved from: https://link.springer.com/book/9781430242871

19. Maksymchuk, S., Kabak, L., & Moroz, B. (2019). Using of the modern data mining technics in customs of Ukraine. *Systems and Technologies*, 2(58), 33–49. https://doi.org/10.32836/2521-6643-2019-2-58-2

20. Khaund, T., Hussain, M. N., Shaik, M., & Agarwal, N. (2020, October). Telegram: Data collection, opportunities and challenges. In Annual international conference on information management and big data (pp. 513–526). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-76228-5_37

21. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Longman Publishing Co., Inc., USA. Retrieved from: https://dl.acm.org/doi/abs/10.5555/579257

22. Oleksiichuk, Y. F., Olkhovska, O. V., Olkhovsky, D. M., & Orlova, D. I. (2023). Design and development of a web service for generating and sending pdf documents. *Systems and Technologies*, *65*(1), 39–45. https://doi.org/10.32782/2521-6643-2023.1-65.5

23. Walls, C. (2022). Spring in action. Simon and Schuster.

24. Krnac, L. (2015). Tasks and Scheduling. In: Pivotal Certified Spring Enterprise Integration Specialist Exam. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-0793-2_1

25. Juneau, J. (2017). Exceptions and Logging. In: Java 9 Recipes. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-1976-8_9

26. Dos Santos, J. S., Azevedo, L. G., Soares, E. F., Thiago, R. M., & da Silva, V. T. (2020). Analysis of Tools for REST Contract Specification in Swagger/OpenAPI. In ICEIS (2) (pp. 201–208). https://doi.org/10.5220/0009381202010208

Дата надходження статті: 21.10.2025
Дата прийняття статті: 10.11.2025
Опубліковано: 30.12.2025