UDC 004.415:638.1

DEVELOPMENT OF AN EVENT-DRIVEN MICROSERVICES-BASED SYSTEM FOR BEEKEEPING MANAGEMENT

Batii K.I.
e-mail: kateryna.batii@gmail.com
Kharkiv National University of Radio Electronics, Systems Engineering dept.
Kharkiv, Ukraine

The main goal of this work is to study and develop individual components of the distributed and scalable system for beekeeping management based on effective approaches, patterns of microservices and event-driven architecture. By leveraging such architecture, the system ensures scalability and real-time data processing. The system under development refers to a system for monitoring, managing and analyzing beekeeping data. Such a system could be used for enhancing productivity and profitability, and improving the quality of life for bees both at amateur apiaries and at industrial enterprises.

Beekeeping is a branch of agriculture that is not only extremely interesting but also good for the environment. It is an important industry that plays a significant role in the pollination and the production of bee products.

In recent years, Ukraine has confidently held the top spot in Europe for honey production and is among the top three global leaders [4]. According to 2020 statistics [4], there were about 400,000 beekeepers in the country, managing over 3 million bee colonies. As a result, Ukraine collected tens of thousands of tons of honey per year, most of which were exported [4].

A European study investigated the adoption of digital beehive monitoring technology through a survey of beekeepers across 18 countries [2, p. 1]. The beekeepers who were already monitoring their apiaries using digital technologies reported stronger expected benefits from using such technologies than the beekeepers who had not yet implemented any kind of digital hive monitoring [2, p. 6]. By digitally tracking and analyzing their hives, they noticed easier management, time and cost savings, improved bee colony health, and reduced winter bee loss [2, p. 6], which consequently will increase profitability [2, p. 8].

Therefore, the development and integration of such a management system into Ukraine's beekeeping is relevant and beneficial for replacing paperwork with digital accounting, saving time by automatically analyzing the data (tracked apiary and hives' info, bee health, honey production, historical data for prediction, and more), providing the hive check schedule, assisting users, notifying them about potential risks, etc. This innovation could be the main tool in the work of beekeepers, without which their work would be disorganized and unproductive, and in some cases impossible. Especially for inexperienced beekeepers and hobbyists, such a tool could help to speed up their learning and avoid irreparable mistakes in working with bees, by guiding them with the application.

Initially, the system was planned to be implemented with a traditional monolithic architecture – a single, unified system [3] with its modules bundled

together into a single codebase. This pattern is convenient in the early stages of the application development life cycle, providing easier development, testing, and simple interactions of code modules since they occur locally [3]. The addition of new functionality, which made the monolith bloated; the need to use other programming languages arising from a lack of flexibility, as the monolith is constrained by the technologies that are already in use; all led to the rejection of the approach and made a redesign a necessity. Therefore, for applications that are constantly evolving, and programming teams that are expanding, it is better to develop business solutions based on microservices.

Microservices decouple major business into separate independent services with their own business logic and database for a specific goal [3]. Examination of the new architecture raised questions about maintaining and deploying microservices, their communication with each other, and user interaction. Consequently, several reasonably useful basic patterns and components were found, which highlight the architecture's flexible, scalable, and reliable approach to development.

To resolve the challenge of dynamically locating and managing services, a service registry is used and it is one of the core components in microservice infrastructure that enables dynamic microservice registration and lookup, ensuring seamless communication between services.

Since the system now consists of many microservices and a considerable amount of their instances, it is preferable to have the user interact with an API gateway that acts as a single-entry point and routes requests to the appropriate microservice [1] using a service registry. This pattern ensures stability, as microservice instances may change their locations, which would otherwise require frequent updates of hardcoded addresses on the frontend. It also can handle user authentication and load balancing, i.e the pattern of forwarding incoming network traffic to several instances to avoid any server from getting overloaded [1].

Microservices can request data for their business processes from other services, and their communication can be synchronous in which one microservice calls another directly and waits for the response, or asynchronous, using message queue systems to process multiple requests at the same time. For this project, it was decided to modify existing microservices by applying the principles of event-driven architecture, where inter-service communication is facilitated through an asynchronous messaging queuing system or, in this case, event streaming platform Apache Kafka [1]. Such architecture offers distinct advantages over synchronous communication, especially in cases like real-time hive monitoring using IoT devices, and detection with notifications of environmental risks, where immediate data flow and user actions are crucial. While synchronous communication can cause delays by waiting for the responses before proceeding the function.

Apache Kafka works by storing streams of records (events) in topics, which can be published by producers and consumed by services [5]. This approach maximizes the decoupling of microservices and improves system reliability. Also, it was chosen for its high performance and low latency [5].

Some of the beekeeping management microservices could be considered high-demand and require high-availability, therefore scalability should be ensured. Depending on factors like the number of users or during some seasons, certain microservices may be less demanded. To avoid wasting resources on keeping all the services running all the time, leveraging containerization using Docker and Kubernetes orchestration, the system will automatically adjust the number of replicas, i.e the number of microservice's instances, based on resource usage or traffic demand. Minimizing the number of replicas during periods of low demand will optimize resource usage and reduce operation costs.

The adoption of a microservices architecture optimizes sustainability (they can be deployed in the cloud, protecting data from the impact of unpredictable situations), robustness, and scalability, as it allows separate deployment and scaling of individual services in relation to demand. The beauty of the microservices architecture lies in its flexibility, allowing developers to model and configure the system with patterns in a way that suits their needs.

Beekeeping is the major economic activity in Ukraine, and this system could make Ukrainian beekeeping industry stronger by attracting more people to beekeeping by simplifying the process for beginners, and providing useful tools for professionals. The more bees there are, the healthier and more balanced the ecosystem becomes. With more beekeepers, the bee population will grow, leading to a positive impact on the environment of our country, especially in these times.

List of references:

1. 9 essential components of a production microservice application. Alex Xu. URL: https://blog.bytebytego.com/p/ep121-9-essential-components-of-a (date of access: 19.02.2025).

2. European beekeepers' interest in digital monitoring technology adoption for improved beehive management / W. Verbeke et al. Computers and electronics in agriculture. 2024. Vol. 227. P. 109556. URL: https://doi.org/10.1016/j.compag.2024.109556 (date of access: 03.03.2025).

3. Microservices vs. monolithic architecture. URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith (date of access: 19.02.2025).

4. Beekeeping as a business: how to invest properly to make a profit. URL: https://www.oschadbank.ua/blog/bdzhilnyctvo-yak-biznes-yak-pravylno-investuvaty-shchob-otrymaty-prybutok (date of access: 20.02.2025).

5. RabbitMQ vs Kafka – Difference Between Message Queue Systems – AWS Inc. URL: https://aws.amazon.com/compare/the-difference-between-rabbitmq-and-kafka/?nc1=h_ls (date of access: 20.02.2025).