# РОЗДІЛ ІІ. ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.054
DOI https://doi.org/10.26661/2786-6254-2022-2-05

## ПОРІВНЯЛЬНЕ ТЕСТУВАННЯ ЗАСТОСУНКІВ, РОЗРОБЛЕНИХ НА ОСНОВІ МОНОЛІТНОЇ ТА МІКРОСЕРВІСНОЇ АРХІТЕКТУР

**Лимаренко Ю. О.**
*кандидат технічних наук, доцент,*
*доцент кафедри електроніки, інформаційних систем та програмного забезпечення*
*Запорізький національний університет*
*вул. Жуковського, 66, Запоріжжя, Україна*
*orcid.org/0000-0002-1643-6939*
*yalymarenko@gmail.com*

**Попівщий В. І.**
*кандидат фізико-математичних наук, доцент,*
*доцент кафедри електроніки, інформаційних систем та програмного забезпечення*
*Запорізький національний університет*
*вул. Жуковського, 66, Запоріжжя, Україна*
*orcid.org/0000-0002-2673-987X*
*pvi@zsea.edu.ua*

**Новак В. В.**
*магістрант кафедри електроніки, інформаційних систем та програмного забезпечення*
*Запорізький національний університет*
*вул. Жуковського, 66, Запоріжжя, Україна*
*orcid.org/0000-0001-5371-4849*
*vit.nov29@gmail.com*

**Міхайлуца О. М.**
*кандидат технічних наук, доцент,*
*доцент кафедри електроніки, інформаційних систем та програмного забезпечення*
*Запорізький національний університет*
*вул. Жуковського, 66, Запоріжжя, Україна*
*orcid.org/0000-0003-2935-7997*
*elenamikhaylutsa7@gmail.com*

**Скрипник І. А.**
*кандидат фізико-математичних наук, доцент,*
*доцент кафедри електроніки, інформаційних систем та програмного забезпечення*
*Запорізький національний університет*
*вул. Жуковського, 66, Запоріжжя, Україна*
*orcid.org/0000-0002-9175-2683*
*sia@zsea.edu.ua*

***Ключові слова:*** *монолітна архітектура, мікросервісна архітектура, пропускна здатність, час відгуку.*

На сьогодні можна виділити два основні підходи до створення вебзастосунків – це використання монолітної та мікросервісної архітектур. Питання, який з цих двох підходів є кращим, залишається відкритим. Відповідь на нього є неоднозначною і залежить від багатьох чинників. Тому проблема порівняння цих двох архітектур на основі тестування за різними показниками є досить актуальною. Метою цієї роботи є порівняння монолітної та мікросервісної архітектур на основі аналізу результатів тестування продуктивності та пропускної здатності розроблених вебзастосунків. Для того щоб мати можливість порівнювати між собою два архітектурні підходи, треба мати два різні застосунки, які б вирішували одні й ті ж самі бізнес-задачі, але один повинен бути реалізований на базі монолітної архітектури, а інший – із застосуванням мікросервісів. Як спільну бізнес-задачу було вибрано здійснення та опрацювання онлайн-замовлень продуктів з метою подальшої адресної доставки цих замовлень кур'єрами. Серверні частини обох застосунків були реалізовані на мові Java за допомогою Java Spring Framework та Java Spring Cloud. Як СУБД було використано PostgreSQL. У застосунку, який побудовано на основі мікросервісної архітектури, взаємодію основних сервісів з базами даних реалізовано у відповідності до принципу «Одна база даних на один сервіс». Для навантажувального тестування та визначення продуктивності обох застосунків було вибрано інструменти JMeter та Gatling. Результати тестування в більшості тестових сценаріїв продемонстрували невелику перевагу монолітної архітектури над мікросервісною. Але для випадку, коли запити стосуються декількох сервісів і, відповідно, декількох баз даних одночасно, можна спостерігати менший час відгуку у застосунку, побудованому за допомогою мікросервісів.

---

# COMPARATIVE TESTING OF APPLICATIONS DEVELOPED BASED ON MONOLITHIC AND MICROSERVICES ARCHITECTURES

**Lymarenko Y. O.**
*Candidate of Technical Sciences, Associate Professor,*
*Associate Professor at the Department of Electronics,*
*Information Systems and Software Engineering*
*Zaporizhzhia National University*
*Zhukovskoho str., 66, Zaporizhzhia, Ukraine*
*orcid.org/0000-0002-1643-6939*
*yalymarenko@gmail.com*

**Popivshchyi V. I.**
*Candidate of Physical and Mathematical Sciences, Associate Professor,*
*Associate Professor at the Department of Electronics,*
*Information Systems and Software Engineering*
*Zaporizhzhia National University*
*Zhukovskoho str., 66, Zaporizhzhia, Ukraine*
*orcid.org/0000-0002-2673-987X*
*pvi@zsea.edu.ua*

**Novak V. V.**
*Master Student at the Department of Electronics, Information Systems and Software Engineering*
*Zaporizhzhia National University*
*Zhukovskoho str., 66, Zaporizhzhia, Ukraine*
*orcid.org/0000-0001-5371-4849*
*vit.nov29@gmail.com*

**Mikhailutsa O. M.**
*Candidate of Technical Sciences, Associate Professor,*
*Associate Professor at the Department of Electronics,*
*Information Systems and Software Engineering*
*Zaporizhzhia National University*
*Zhukovskoho str., 66, Zaporizhzhia, Ukraine*
*orcid.org/0000-0003-2935-7997*
*elenamikhaylutsa7@gmail.com*

**Skrypnyk I. A.**
*Candidate of Physical and Mathematical Sciences, Associate Professor,*
*Associate Professor at the Department of Electronics,*
*Information Systems and Software Engineering*
*Zaporizhzhia National University*
*Zhukovskoho str., 66, Zaporizhzhia, Ukraine*
*orcid.org/0000-0002-9175-2683*
*sia@zsea.edu.ua*

***Key words:*** *monolithic architecture, microservices architecture, throughput, response time.*

Nowadays, there are two basic approaches to web application development: monolithic and microservices architectures. The question of which of these two approaches is better remains open. The answer is controversial and depends on many factors. Therefore, the problem of comparing these two architectures based on various indicators testing is quite actual. The aim of the paper is to compare monolithic and microservices architectures. We use analysis of performance and throughput testing of developed web applications. In order to be able to compare two architectural approaches, it is necessary to have two different applications that would solve the same business problems, but one should be implemented on the basis of a monolithic architecture, and the other should be implemented with the use of microservices. Implementation and processing of online product orders with the aim of further address delivery of these orders by couriers was chosen as a business task. The server parts of both applications were developed on Java platform using Java Spring Framework and Java Spring Cloud. PostgreSQL was used as a DBMS. The interaction of main services with databases is implemented in according to the principle "One database for one service" in the microservice application. JMeter and Gatling tools were chosen for load testing and performance determination of both applications. The results in most test scenarios showed a slight advantage of the monolithic architecture over the microservices architecture. But for the case of requests with multi services and multi databases at the same time, it is observed a shorter response time for the microservice application.

**Вступ**

Починаючи приблизно з 2014 року в IT-індустрії спостерігається неабиякий інтерес до підходу, який передбачає розробку програмного забезпечення на основі мікросервісної архітектури. Точніше, окремі спроби застосування цього підходу спостерігались і раніше, ще з початку 2000-х років, але з появою Docker-контейнерів у 2013 році стало зрозумілим, що у цього підходу велике майбутнє. Цей факт було зафіксовано, зокрема, і в статті Мартіна Фаулера та Джеймса Льюіса [1], в якій термін «мікросервісна архітектура» визначено як стиль розробки програмних застосунків у вигляді набору сервісів, які розгортаються незалежно.

На сьогодні ми вже спостерігаємо картину, коли великі корпорації успішно використовують мікросервісну архітектуру для організації своїх бізнес-процесів. Це частково або повністю стосується таких компаній, як Netflix, Amazon, eBay, Uber, Tencent, Zalando, Spotify, Airbnb, LinkedIn, Twitter, Groupon, Coca-Cola та ін.

Сфера застосування цього підходу стає дедалі ширшою. Так, у роботах [2; 3] наведено огляд публікацій, присвячених тим можливостям, які дає поєднання мікросервісного підходу та блокчейн-технологій. Цікавою та корисною з практичної точки виглядає синергія Інтернету речей та мікросервісів [4]. Ну і, звичайно, там, де виникає необхідність опрацьовувати великі обсяги даних, застосування мікросервісів дозволяє вирішити велику кількість проблем, які мали місце у разі застосування монолітної архітектури застосунку [4; 5].

З іншого боку, як і будь-яка інша технологія, мікросервіси мають і свої слабкі місця. Одним з цих слабких місць є безпека. Цій темі присвячено досить багато літератури. Змістовний огляд проблеми безпеки мікросервісних застосунків можна знайти в роботах [6; 7].

Оскільки мікросервіси позиціонуються як альтернатива монолітному підходу, то, звичайно, виникає купа питань на кшталт: чи варто вже працюючу систему переносити на рейки монолітної архітектури, якщо так, то яким чином це робити з мінімальними витратами та ризиками, що очікувати від системи з точки зору швидкості опрацювання запитів та пропускної здатності після такого переносу, і т.п. Частково відповіді на ці питання можна знайти в літературі. Але наявні дослідження демонструють різноманітність в оцінках продуктивності двох різних архітектур. Так, у роботі [8] авторами проведено порівняльний аналіз двох вебзастосунків, реалізованих на основі двох різних архітектур, і висновок щодо більш ефективного використання обчислювальних ресурсів апаратного забезпечення та продуктивності самого застосунку робиться на користь мікросервісної архітектури. Дослідження, проведене авторами роботи [9], спрямоване як на порівняння монолітної архітектури з мікросервісною, так і на порівняння засобів розробки для серверної частини, а саме Java та C#.Net. Результати тестування, проведеного авторами, свідчать, що у разі розгортання застосунків на одній обчислювальній машині монолітна архітектура з точки зору продуктивності має перевагу над мікросервісною. Аналогічного висновку дійшли й автори роботи [10], в якій описано різні тестові сценарії. І деякі з них продемонстрували невелику перевагу монолітної архітектури над мікросервісною, інші ж суттєвої різниці не виявили.

У зв'язку з такою неоднозначністю в оцінках, наведених у різних роботах, можна фіналізувати, що питання порівняння двох архітектур продовжує залишатись відкритим. Тому будь-які дослідження в цьому напрямі будуть тільки збільшувати експериментальну базу з метою отримання більш надійного підгрунтя для прийняття обґрунтованого рішення у разі вибору того чи іншого архітектурного підходу. Власне, метою цієї роботи як раз і є порівняння монолітної та мікросервісної архітектури на основі аналізу результатів тестування продуктивності та пропускної здатності розроблених застосунків за різними тестовими сценаріями.

**Опис застосунків, що використовувались для порівняння двох архітектур**

Для того щоб мати можливість порівнювати між собою два архітектурні підходи, треба мати два різних застосунки, які вирішують одні й ті ж самі бізнес-задачі, але один повинен бути реалізований на базі монолітної архітектури, а інший – із застосуванням мікросервісів. Як бізнес-задачу для реалізації було вибрано розробку вебзастосунку, який би надавав таку актуальну останніми, ковідівськими, роками послугу, як онлайн-замовлення продуктів з метою подальшої адресної доставки цих замовлень кур'єрами.

Основні функціональні вимоги до застосунків були сформульовані таким чином: 1) у системі повинні бути передбачені такі ролі: адміністратора, замовника продуктів, продавця та кур'єра; 2) для замовників повинна бути наявною можливість замовити, оплатити та переглянути історію замовлень; 3) кур'єр повинен мати можливість переглядати замовлення та приймати їх на виконання; 4) продавці повинні мати змогу оновлювати інформацію про наявні продукти.

Серверні частини обох застосунків були реалізовані на мові Java за допомогою Java Spring Framework та Java Spring Cloud. Як СУБД було використано PostgreSQL. Клієнтська частина реалізована за допомогою Angular Framework. Для оновлення міграцій даних використовувався

Flyway Migration. Система захисту була побудована з використанням бібліотеки JWT та вбудованої системи Java Spring Security.

У таблиці 1 представлено мікросервіси, кожен з яких працює на своєму окремому порті. Поряд з мікросервісами вказано також бази даних, які використовуються мікросервісами як власне сховище даних. Як видно з таблиці, основні мікросервіси працюють з власною базою даних, тобто взаємодію майже всіх мікросервісів з базами даних реалізовано у відповідності до принципу «Одна база даних на один сервіс».

Таблиця 1

**Мікросервіси та бази даних, з якими вони взаємодіють**

| Мікросервіси | Бази даних |
|---|---|
| users-service | userdb |
| products-service | productdb |
| orders-service | orderdb |
| payment-service | paymentdb |
| delivery-service | deliverydb |
| security-service | userdb |
| cloud-config | |
| discrovery | |
| cloud-gateway | |
| hystrix-dashboard | |

Розгортання застосунку, побудованого на основі монолітної архітектури, відбувалось за допомогою Maven. Для розгортання застосунку, побудованого за допомогою мікросервісної архітектури, було використано Docker та Kubernates.

**Методика та результати порівняння**

Для порівняти двох тестових застосунків було вибрано такі показники, як: швидкість відповіді на велику кількість запитів, кількість провалених запитів, середнє значення відповіді, коефіцієнт помилок, пропускна здатність. Для навантажувального тестування та визначення продуктивності обох застосунків було вибрано інструменти JMeter і Gatling. При цьому обидва застосунки було розгорнуто у локальному середовищі.

Під час тестування як показник продуктивності було використано пропускну здатність, яка обчислюється JMeter як кількість запитів, оброблених сервером, поділена на загальний час обробки запитів у секундах. Час вимірюється від початку першого запиту до кінця останнього запиту. Сюди входять будь-які інтервали між запитами, оскільки вони мають відображати навантаження на сервер. Оскільки тести призначені виключно для збору інформації про продуктивність ресурсів на стороні сервера, то час, необхідний для обробки інформації на стороні клієнта, не враховується.

Під час тестування генерувались різні типи запитів, які були адресовані різним мікросервісам. Кількість запитів у кожному з тестових сценаріїв становила 1000. Так, на рисунку 1 представлено результати обчислення пропускної здатності монолітного та мікросервісного застосунків для 4 видів запитів. Перший, третій та четвертий види запитів пов'язані з отриманням даних, другий – з додаванням. Але кожен з цих 4 видів запитів оброблювався лише одним конкретним сервісом і, відповідно, ініціював звернення лише до бази даних саме цього сервісу.

Як можна побачити з діаграми, у всіх випадках мікросервісна архітектура демонструє нижчу пропускну здатність ніж монолітна. Мінімальне відхилення становить 8,22%, максимальне – 21,93%. Однією з основних причин такої різниці

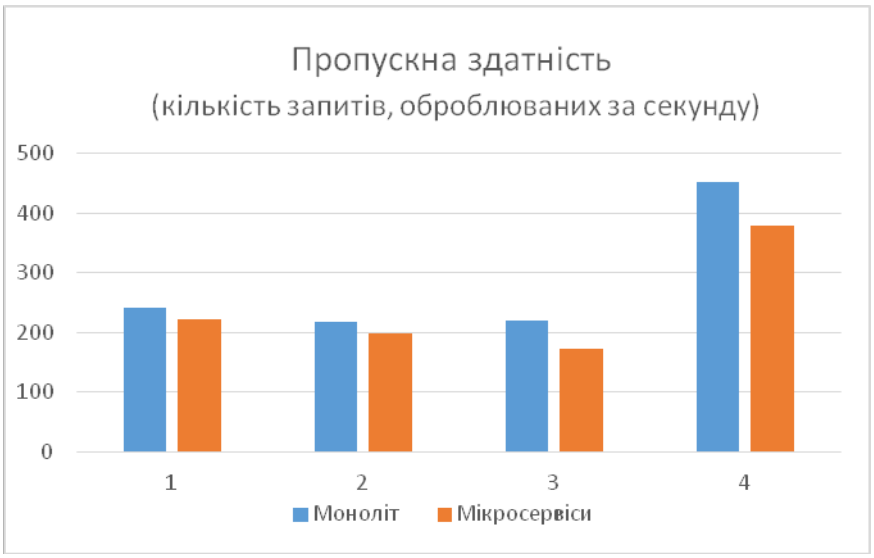


**Рис. 1. Порівняння пропускної здатності застосунків з монолітною та мікросервісною архітектурами**

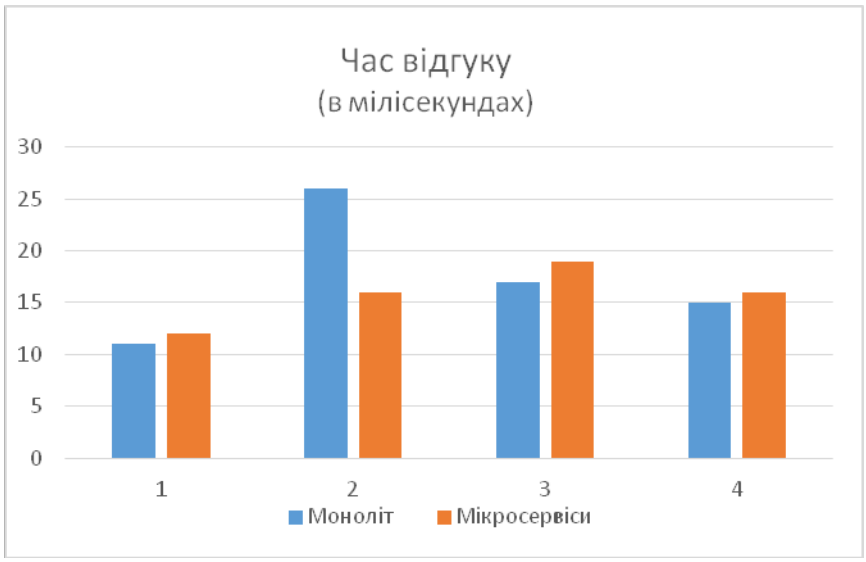


**Рис. 2. Порівняння середнього часу відгуку застосунків з монолітною та мікросервісною архітектурами**

в показниках пропускної здатності є те, що у випадку мікросервісного застосунку запити до сервісу передаються через API Gateway, що створює накладні витрати на зв'язок, що погіршує продуктивність.

За допомогою Gatling проводилось обчислення часу відгуку обох систем, який визначається цим інструментом як проміжок часу з моменту відправки запиту до сервера до моменту, коли буде отримана відповідь, або буде повернуто помилку. На рисунку 2 продемонстровано порівняння середнього часу відгуку обох застосунків також для 4 різних видів запитів. Перший, третій та четвертий види запиту йдуть на обробку до одного сервісу (коже вид запиту до свого сервісу) і, відповідно, ініціюють звернення до однієї бази даних. Другий вид зипиту є комплексним і в його обробці беруть участь одночасно два сервіси і, відповідно, дві бази даних.

Як видно з наведеної діаграми, у випадках, коли запити оброблюються одним сервісом, час відгуку менший у застосунку з монолітною архітектурою, причому мінімальна різниця становить 6,67%, а максимальна – 11,76%. Для запитів, які оброблюються двома сервісами, середній час відгуку у застосунку з мікросервісною архітектурою менший порівняно з монолітним застосунком на 38,46%. Такий результат, імовірно, пов'язаний з тим, що в мікросервісній архітектурі різні сервіси можуть обробляти запити паралельно, тоді як у застосунках з монолітною архітектурою це складніше реалізувати.

**Висновки**

Порівняльний аналіз результатів тестування двох застосунків, реалізованих за допомогою мікросервісної та монолітної архітектур, показав, що за такими показниками, як пропускна здатність та час відгуку, монолітна архітектура здебільшого демонструє кращі результати ніж мікросервісна. Перевага, продемонстрована тестовими застосунками, у середньому становить 10–15%. Водночас у тому випадку, коли йдеться про запити, в обробці яких беруть участь декілька сервісів, можемо бачити, що можливість паралельної обробки, яку надають мікросервіси, дозволяє їм швидше обробляти такі запити, ніж це має місце в застосунках з монолітною архітектурою.

Загалом, як це і ставилось за мету у формуванні підходу, що базується на використанні мікросервісів для створення вебзастосунків, головні переваги цієї парадигми полягають не у швидкості роботи, а у швидкості реагування на зміни бізнес-вимог, у можливості незалежного та швидкого розгортання, незалежного масштабування, стійкості до відмов, можливості здійснювати розробку за допомогою різних технічних стеків.

## ЛІТЕРАТУРА

1. Fowler M. and Lewis J. "Microservices", 2014. URL: http://martinfowler.com/articles/microservices.html.
2. Santos R., Soares P., Rodrigues E., Maia P.H.M. and Silveira A. How Blockchain and Microservices are Being Used Together: a Systematic Mapping Study, 2022. IEEE/ACM *5th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, 2022, pp. 39–46.

3. Oumoussa I., Faieq S., Saidi R. When Microservices Architecture and Blockchain Technology Meet: Challenges and Design Concepts. *International Conference on Advanced Technologies for Humanity (ICATH)*. 2021. Lecture Notes on Data Engineering and Communications Technologies. Vol. 110, pp. 161–172.
4. Atitallah S.B., Driss M., Ghzela H.B. Microservices for Data Analytics in IoT Applications: Current Solutions, Open Challenges, and Future Research Directions. *Procedia Computer Science*. Volume 207. 2022, pp. 3938–3947.
5. Laigner R. et al. "From a Monolithic Big Data System to a Microservices Event-Driven Architecture". *46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2020, pp. 213–220.
6. Berardi D., Giallorenzo S., Melis A., Prandini M., Mauro J., Montesi F. Microservice Security: A Systematic Literature Review. *Peer J Computer Science.* 2022, 8:e779.
7. Pereira-Vale A., Márquez G., Astudillo H., Fernandez E.B. Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping. *XLV Latin American Computing Conference (CLEI)*. 2019, pp. 1–10.
8. Tapia F., Mora M.Á., Fuertes W., Aules H., Flores E., Toulkeridis T. From Monolithic Systems to Microservices: A Comparative Study of Performance. *Applied Sciences*. 2020, 10(17): 5797.
9. Blinowski G., Ojdowska A., Przybyłek A. Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. *IEEE Access*. 2022, 10, pp. 20357–20374.
10. Al-Debagy O., Martinek P. A Comparative Review of Microservices and Monolithic Architectures. In *Proceedings of the 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, Budapest, Hungary, 21–22 November 2018, pp. 149–154.

## REFERENCES

1. Fowler, M. and Lewis, J. (2014). "Microservices". Retrieved from: http://martinfowler.com/articles/microservices.html.
2. Santos, R., Soares, P., Rodrigues, E., Maia, P.H.M. and Silveira, A. (2022). How Blockchain and Microservices are Being Used Together: a Systematic Mapping Study. 2022 IEEE/ACM 5th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), pp. 39–46.
3. Oumoussa I., Faieq S., Saidi R. (2021). When Microservices Architecture and Blockchain Technology Meet: Challenges and Design Concepts. International Conference on Advanced Technologies for Humanity (ICATH). Lecture Notes on Data Engineering and Communications Technologies. Vol. 110, pp. 161–172.
4. Atitallah, S.B., Driss, M., Ghzela, H.B. (2022). Microservices for Data Analytics in IoT Applications: Current Solutions, Open Challenges, and Future Research Directions. Procedia Computer Science. Volume 207, pp. 3938–3947.
5. Laigner, R. et al. (2020). "From a Monolithic Big Data System to a Microservices Event-Driven Architecture". 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 213–220.
6. Berardi, D., Giallorenzo, S., Melis, A., Prandini, M., Mauro, J., Montesi, F. (2022). Microservice Security: A Systematic Literature Review. Peer J Computer Science. 8:e779.
7. Pereira-Vale, A., Márquez, G., Astudillo, H., Fernandez, E.B. (2019). Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping. XLV Latin American Computing Conference (CLEI), pp. 1–10.
8. Tapia, F., Mora, M.Á., Fuertes W., Aules H., Flores E., Toulkeridis T. (2020). From Monolithic Systems to Microservices: A Comparative Study of Performance. Applied Sciences, 10(17): 5797.
9. Blinowski, G., Ojdowska, A., Przybyłek, A. (2022). Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. IEEE Access, 10, pp. 20357–20374.
10. Al-Debagy, O., Martinek, P. (2018). A Comparative Review of Microservices and Monolithic Architectures. In Proceedings of the IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 21–22 November 2018, pp. 149–154.