SCI-CONF.COM.UA

CURRENT TRENDS IN SCIENTIFIC RESEARCH DEVELOPMENT

PROCEEDINGS OF VI INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE JANUARY 16-18, 2025

BOSTON
2025

CURRENT TRENDS IN SCIENTIFIC RESEARCH DEVELOPMENT

Proceedings of VI International Scientific and Practical Conference

Boston, USA

16-18 January 2025

Boston, USA

2025

UDC 001.1

The 6th International scientific and practical conference "Current trends in scientific research development" (January 16-18, 2025) BoScience Publisher, Boston, USA. 2025. 819 p.

ISBN 978-1-73981-122-8

The recommended citation for this publication is:

Ivanov I. Analysis of the phaunistic composition of Ukraine // Current trends in scientific research development. Proceedings of the 6th International scientific and practical conference. BoScience Publisher. Boston, USA. 2025. Pp. 21-27. URL: https://sci-conf.com.ua/vi-mizhnarodna-naukovo-praktichna-konferentsiya-current-trends-in-scientific-research-development-16-18-01-2025-boston-ssha-arhiv/.

Editor
Komarytskyy M.L.
Ph.D. in Economics, Associate Professor

Collection of scientific articles published is the scientific and practical publication, which contains scientific articles of students, graduate students, Candidates and Doctors of Sciences, research workers and practitioners from Europe, Ukraine and from neighbouring countries and beyond. The articles contain the study, reflecting the processes and changes in the structure of modern science. The collection of scientific articles is for students, postgraduate students, doctoral candidates, teachers, researchers, practitioners and people interested in the trends of modern science development.

e-mail: boston@sci-conf.com.ua

homepage: *https://sci-conf.com.ua*

TABLE OF CONTENTS

AGRICULTURAL SCIENCES

VETERINARY SCIENCES

MEDICAL SCIENCES

PHYSICAL AND MATHEMATICAL SCIENCES

УДК 004.43:004.4'272:004.738.5

ОПТИМІЗАЦІЯ ЕЛЕКТРОННОЇ КОМЕРЦІЇ ЗА ДОПОМОГОЮ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Хаджинов Руслан Валерійович,
Студент
Чудний Тарас Евгенович,
Технічний енеджер ГО «ОЛ ТУГЕЗЕР ДЖОБС»
Чуніхін Артем Сергійович,
аспірант
Селівьорстова Тетяна Віталіївна,
к.т.н., доцент
Український державний університет науки і технологій
м. Дніпро, Україна

Анотація: У статті розглянуто мікросервісну архітектуру як ефективний підхід до розробки високомасштабованих додатків електронної комерції. Проаналізовано ключові переваги цього підходу, такі як масштабованість, стійкість до збоїв, гнучкість у виборі технологій та швидке впровадження нових функцій. Окрему увагу приділено шаблонам мікросервісної архітектури, інфраструктурним рішенням, а також практичним аспектам реалізації системи з використанням сучасних інструментів Docker, Kubernetes, CI/CD і хмарних платформ. Результати дослідження демонструють, що впровадження мікросервісної архітектури сприяє підвищенню ефективності роботи додатків і забезпечує гнучкість бізнес-процесів.

Ключові слова: Мікросервісна архітектура, електронна комерція, масштабованість, Kubernetes, Docker, CI/CD, інфраструктурні рішення, шаблони архітектури, хмарні платформи.

Вступ. У сучасних умовах стрімкого розвитку цифрових технологій і зростання обсягу даних, електронна комерція стикається з необхідністю забезпечення високої масштабованості, надійності та гнучкості програмних рішень. Традиційні монолітні архітектури часто не здатні ефективно

задовольнити ці вимоги, що обумовлює актуальність впровадження мікросервісної архітектури.

Мікросервісна архітектура передбачає розділення програмного забезпечення на незалежні, дрібні компоненти (мікросервіси), кожен з яких виконує конкретну бізнес-функцію і може розгортатися, масштабуватися та оновлюватися окремо. Такий підхід дозволяє забезпечити високу стійкість до збоїв, гнучкість у виборі технологій, спрощення процесу розробки й обслуговування системи.

Особливо важливим застосування мікросервісної архітектури є в сфері електронної комерції, де критично важливими є безперервна доступність сервісів, швидка обробка великої кількості транзакцій та можливість масштабування системи відповідно до зростаючих потреб бізнесу. Впровадження мікросервісних підходів дозволяє значно підвищити ефективність розробки та експлуатації таких систем.

Метою цієї статті є дослідження шаблонів мікросервісної архітектури, їх переваг та недоліків, а також аналіз практичного застосування мікросервісного підходу для створення високомасштабованих додатків електронної комерції.

Переваги мікросервісної архітектури

1. Масштабованість. Мікросервіси дозволяють масштабувати окремі компоненти системи залежно від навантаження. Це дає змогу ефективно використовувати ресурси, знижуючи витрати на інфраструктуру. Наприклад, у разі підвищеного попиту на певну функціональність можна збільшити кількість її екземплярів без впливу на інші частини системи.

2. Висока стійкість до збоїв. Оскільки кожен мікросервіс функціонує незалежно, відмова одного сервісу не впливає на роботу інших. Це підвищує загальну надійність системи та мінімізує ризики простою. Впровадження автоматичних механізмів відновлення, таких як Hystrix, дозволяє ще більше підвищити стійкість до збоїв.

3. Гнучкість у виборі технологій. Кожен мікросервіс може розроблятися на різних мовах програмування та використовувати різні технології, що

дозволяє командам обирати оптимальні рішення для конкретних задач. Це сприяє швидшому впровадженню нових технологій і покращенню продуктивності розробки.

4. Прискорення розробки та впровадження. Незалежність мікросервісів дозволяє різним командам одночасно працювати над окремими компонентами системи. Це зменшує час на розробку, тестування та впровадження нових функцій, що забезпечує швидший вихід продукту на ринок.

5. Полегшене обслуговування та масштабування команд. Завдяки чіткій структурі мікросервісів великі команди розробників можуть бути поділені на невеликі автономні групи, що спрощує управління проектом і підвищує ефективність розробки.

6. Покращена безпека. Ізоляція мікросервісів забезпечує кращий контроль над доступом до даних і ресурсів. Це дозволяє впроваджувати більш гнучкі політики безпеки та зменшує ризики витоку інформації.

Таким чином, мікросервісна архітектура забезпечує гнучке, масштабоване і надійне середовище для розробки та впровадження програмних рішень, що є критично важливим для сучасних додатків електронної комерції.

Шаблони мікросервісної архітектури. Шаблони мікросервісної архітектури є стандартними підходами до проєктування та реалізації мікросервісних систем. Вони допомагають вирішувати типові проблеми, що виникають під час розробки складних програмних продуктів, забезпечують структурованість і передбачуваність архітектурних рішень. Завдяки використанню перевірених шаблонів можна скоротити час розробки, підвищити якість коду та уникнути поширених помилок. Крім того, шаблони сприяють стандартизації підходів у команді розробників і забезпечують гнучкість та масштабованість системи.

1. База даних для кожного мікросервісу. Цей шаблон передбачає надання кожному мікросервісу окремої бази даних, що забезпечує незалежність та ізоляцію даних. Такий підхід дозволяє кожному мікросервісу обирати оптимальну модель зберігання даних та мінімізує ризики конфліктів при

оновленнях.

2. API Gateway. API Gateway виступає єдиною точкою входу для клієнтських запитів, що забезпечує маршрутизацію запитів до відповідних мікросервісів. Це дозволяє централізовано керувати автентифікацією, обмеженням доступу та кешуванням.

3. Сага. Сага є шаблоном для управління розподіленими транзакціями між мікросервісами. Вона реалізується через серії локальних транзакцій з компенсаційними діями у випадку збоїв. Сага може бути реалізована через оркестрацію або хореографію.

4. CQRS (Command Query Responsibility Segregation). Цей шаблон розділяє операції запису (команди) і читання (запити), що дозволяє оптимізувати обидва процеси окремо. Це підвищує продуктивність і гнучкість у роботі з даними.

5. Асинхронна комунікація через повідомлення. Асинхронна взаємодія між мікросервісами реалізується через черги повідомлень (наприклад, RabbitMQ або Kafka). Це знижує навантаження на систему та забезпечує кращу стійкість до збоїв.

Ці шаблони дозволяють ефективно будувати масштабовані, стійкі та гнучкі програмні рішення, що відповідають вимогам сучасних бізнес-задач.

Інфраструктурні рішення для мікросервісів. Ефективна робота мікросервісної архітектури потребує відповідної інфраструктури, яка забезпечить надійне розгортання, масштабування і підтримку мікросервісів. Сучасні технології пропонують різноманітні рішення для цих задач.

1. Контейнеризація. Контейнеризація дозволяє ізолювати мікросервіси в окремі контейнери, що включають весь необхідний для роботи програмний стек. Найпопулярнішим інструментом є Docker, який забезпечує легке створення, розгортання і управління контейнерами. Це дозволяє значно знизити залежність від середовища розробки та спростити процес масштабування.

2. Оркестрація контейнерів. Для управління великою кількістю контейнерів використовуються системи оркестрації, такі як Kubernetes.

Kubernetes автоматизує розгортання, масштабування, оновлення та моніторинг контейнеризованих додатків. Він забезпечує балансування навантаження, відновлення після збоїв і безперервну інтеграцію.

3. Хмарні платформи. Хмарні сервіси, такі як Amazon Web Services (AWS), Google Cloud Platform (GCP) і Microsoft Azure, пропонують інфраструктуру для розгортання мікросервісних додатків. Вони забезпечують автоматичне масштабування, високий рівень доступності та зручні інструменти для управління ресурсами.

4. CI/CD (Безперервна інтеграція та доставка). Інструменти CI/CD, такі як Jenkins, GitLab CI/CD, CircleCI, забезпечують автоматизацію процесів збирання, тестування і розгортання мікросервісів. Це дозволяє швидко впроваджувати нові функції і мінімізувати ризики помилок у виробничому середовищі.

5. Системи моніторингу та логування. Для забезпечення стабільної роботи мікросервісів важливе постійне моніторинг і логування. Інструменти такі як Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana) допомагають відстежувати стан системи, аналізувати помилки і прогнозувати навантаження.

6. Секрети та управління конфігураціями. Інструменти для управління конфігураціями і секретами, такі як HashiCorp Vault, Kubernetes ConfigMaps і Secrets, дозволяють безпечно зберігати та керувати конфіденційною інформацією, що є важливою для захисту даних і безпеки додатків.

Впровадження цих інфраструктурних рішень дозволяє забезпечити стабільну, безпечну і масштабовану роботу мікросервісних додатків, що є критично важливим для досягнення бізнес-цілей у сфері електронної комерції.

Практична реалізація. Розробка високомасштабованого додатку електронної комерції на основі мікросервісної архітектури передбачає комплексний підхід до вибору технологій, розробки компонентів і розгортання системи.

1. Вибір мови програмування та фреймворку. Для розробки мікросервісів

було обрано платформу .NET завдяки її потужним можливостям для створення високопродуктивних веб-додатків. ASP.NET Core надає вбудовану підтримку для розробки RESTful API, що дозволяє легко інтегрувати окремі сервіси в єдину систему.

2. Проєктування архітектури додатку. Архітектура додатку передбачає розділення на окремі мікросервіси, кожен з яких виконує конкретну бізнес-функцію:

– Сервіс управління замовленнями — обробка замовлень клієнтів.

– Сервіс інвентаризації — управління товарами на складі.

– Платіжний сервіс — обробка транзакцій і платежів.

– Сервіс доставки — управління логістикою та відстеженням доставок.

3. Розгортання за допомогою Docker і Kubernetes. Для ізоляції і запуску мікросервісів використовується Docker, що дозволяє створювати легкі та портативні контейнери. Kubernetes забезпечує автоматичне масштабування, балансування навантаження та моніторинг роботи контейнерів, що гарантує стабільність і доступність додатку.

4. Організація комунікації між мікросервісами. Для забезпечення ефективної взаємодії між мікросервісами застосовуються як синхронні (HTTP REST API), так і асинхронні (черги повідомлень на основі RabbitMQ) механізми комунікації. Це забезпечує гнучкість і стійкість системи до збоїв.

5. CI/CD процеси. Автоматизація процесу розробки, тестування і розгортання здійснюється за допомогою GitLab CI/CD. Це дозволяє автоматично інтегрувати нові зміни в коді, тестувати їх і розгортати в середовищі розробки або продакшені.

6. Моніторинг та логування. Для моніторингу продуктивності додатку використовуються Prometheus і Grafana, які забезпечують збір і візуалізацію метрик. ELK Stack застосовується для аналізу логів і виявлення помилок.

7. Безпека та управління конфігураціями. Для безпечного зберігання конфіденційних даних і керування конфігураціями використовується HashiCorp Vault і Kubernetes Secrets. Це гарантує надійний захист даних і безпеку додатку.

8. Хмарна інфраструктура. Додаток розгортається на хмарній платформі AWS, що забезпечує автоматичне масштабування, високий рівень доступності та резервне копіювання даних.

Практична реалізація мікросервісної архітектури в додатку електронної комерції дозволяє забезпечити гнучкість, масштабованість і надійність системи, що є критично важливим для успішного ведення бізнесу в умовах сучасного ринку.

Висновки.

Мікросервісна архітектура є ефективним рішенням для створення високомасштабованих і надійних додатків, особливо в сфері електронної комерції.

Завдяки розподіленій структурі, незалежності компонентів і можливості гнучкого масштабування, мікросервіси забезпечують підвищену стійкість до збоїв, швидку розробку та впровадження нових функцій.

Використання шаблонів мікросервісної архітектури, таких як база даних для кожного сервісу, API Gateway, сага та CQRS, сприяє ефективному управлінню бізнес-процесами і забезпечує гнучку інтеграцію між компонентами системи. Інфраструктурні рішення, зокрема контейнеризація за допомогою Docker і оркестрація через Kubernetes, дозволяють досягти високої доступності і стабільності додатку.

Практична реалізація на основі сучасних інструментів CI/CD, систем моніторингу і логування, а також хмарних платформ забезпечує безперервний розвиток і масштабування програмних продуктів. Це дозволяє компаніям швидко реагувати на зміни ринку, зменшувати витрати на інфраструктуру та підвищувати якість обслуговування клієнтів.

Таким чином, впровадження мікросервісної архітектури є стратегічно важливим кроком для підприємств, які прагнуть до стабільного зростання, технологічної гнучкості і високої конкурентоспроможності на сучасному ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Newman, S. (2019). Building Microservices: Designing Fine-Grained Systems (2nd ed.). O'Reilly Media.
2. Richardson, C. (2018). Microservices Patterns: With examples in Java. Manning Publications.
3. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. Communications of the ACM, 59(5), 50-57.
4. Thönes, J. (2015). Microservices. IEEE Software, 32(1), 116-116.
5. Pahl, C., & Jamshidi, P. (2016). Microservices: A Systematic Mapping Study. CLOSER, 137-146.
6. Docker Inc. (2022). Docker Documentation. Retrieved from https://docs.docker.com/
7. Kubernetes Authors. (2022). Kubernetes Documentation. Retrieved from https://kubernetes.io/docs/
8. Nginx Inc. (2022). API Gateway: The Basics. Retrieved from https://www.nginx.com/learn/api-gateway/
9. Amazon Web Services. (2022). AWS Documentation. Retrieved from https://docs.aws.amazon.com/
10. Дмитренко, О. А. (2021). Мікросервісна архітектура та її задачі на прикладах з реального життя. Збірник наукових праць "Інформаційні технології та системи", 2(1), 67-74.
11. Макарова, Д. (2017). Архітектура мікросервісів. Матеріали III Всеукраїнської науково-технічної конференції "Теоретичні та прикладні аспекти радіотехніки і приладобудування", 17-19.
12. Золотарьов, Д. (2021). Мікросервісна архітектура побудови розподілених автоматизованих обчислень високої доступності у хмарній інфраструктурі. Сучасний стан наукових досліджень та технологій в промисловості, (3), 137-146.
13. Нефьодов, Д. А., Удовенко, С. Г., & Чала, Л. Е. (2022). Мікросервісна архітектура системи потокової обробки великих даних. АСУ та

прилади автоматики, 1(178), 50-64.

14. Коваль, А. (2022). Мікросервісна архітектура в розробці програмного забезпечення. Матеріали конференції "Хмарні технології в освіті", 117-120.

15. Щербина, І. С., & Явор, Д. В. (2023). Вплив технології Docker на архітектуру мікросервісів. Наукові записки Державного університету телекомунікацій, (1), 50-57.

16. Киселевич, В., Усата, О., Сікора, Я., Вербівський, Д., & Іванов, Д. (2024). Мікросервісна архітектура: переваги та недоліки її практичного застосування. Журнал "Інформаційні технології", 2(7), 45-53.