

НАПРЯМ 7. КОМП'ЮТЕРНІ НАУКИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

DOI <https://doi.org/10.36059/978-966-397-479-8-123>

РОЗРОБКА МАСШТАБОВАНИХ МІКРОСЕРВІСІВ ЗА ДОПОМОГОЮ SPRING BOOT: ВИКЛИКИ ТА ПЕРЕВАГИ

Биков Р. Г.

*викладач кафедри інформаційних технологій
Міжнародний гуманітарний університет
м. Одеса, Україна*

У сучасному світі динамічний розвиток інформаційних технологій потребує нових підходів до розробки програмного забезпечення. Мікросервісна архітектура – це один із підходів, який дає змогу розробляти масштабовані, гнучкі та стійкі до збоїв системи. У цих умовах технологія Spring Boot стала стандартом для розробки Java-додатків завдяки своїй простоті, широким можливостям та інтеграції з іншими інструментами екосистеми Java. У даній роботі розглядаються основи побудови мікросервісів за допомогою Spring Boot, основні переваги цього підходу, а також виклики, що виникають під час реалізації таких систем.

Мікросервісна архітектура: огляд концепції

Мікросервісна архітектура представляє собою методологію створення програмного забезпечення, який ґрунтується на поділі монолітної системи на невеликі, незалежні сервіси, що виконують окремі бізнес-функції [1]. Основними характеристиками мікросервісів [2] є:

- Модульність: кожен сервіс реалізує чітко визначену функцію.
- Незалежне розгортання: кожен сервіс можна оновлювати чи масштабувати незалежно від інших.
- Гнучкість: сервіси можуть бути реалізовані з використанням різних мов програмування.

Головними викликами при впровадженні мікросервісної архітектури є забезпечення взаємодії між сервісами, управління їхньою масштабованістю, безпекою та стійкістю до збоїв.

Роль Spring Boot у розробці мікросервісів

Spring Boot є фреймворком, створеним для спрощення розробки Java-додатків [3, 4]. Його переваги в контексті мікросервісної архітектури включають:

1. Автоматичне налаштування: Spring Boot надає попередньо налаштовані компоненти для розробки REST API, інтеграції з базами даних і забезпечення безпеки.

2. Швидкий старт: за допомогою Spring Initializr можна створити готовий до роботи шаблон додатка за декілька хвилин.

3. Підтримка Spring Cloud: набір інструментів для оркестрації мікросервісів, таких як Eureka (реєстрація сервісів), Zuul (маршрутизація запитів) та Config Server (централізоване управління конфігураціями).

4. Вбудований веб-сервер: можливість розгортання сервісів без окремого контейнера (наприклад, Tomcat інтегрований із Spring Boot).

Виклики та способи їх вирішення

Попри численні переваги, розробка мікросервісів на Spring Boot супроводжується рядом викликів [5]:

1. Масштабованість.

Масштабування може бути ускладненим через необхідність синхронізації сервісів. Використання Spring Cloud Gateway допомагає впровадити балансування навантаження та оптимізувати обробку запитів.

2. Управління транзакціями.

У розподіленій системі транзакції часто охоплюють кілька сервісів. Для цього використовують шаблон "saga" (Saga Pattern), який Spring Boot підтримує через Spring Cloud Data Flow.

3. Безпека.

Забезпечення автентифікації та авторизації для кожного сервісу може бути трудомістким. Використання Spring Security разом з OAuth2 або JWT спрощує цей процес.

4. Моніторинг та діагностика.

Керування кількома сервісами потребує якісного моніторингу. Інструменти, такі як Spring Boot Actuator та інтеграція з системами APM (наприклад, Prometheus, Grafana), дозволяють отримувати інформацію про стан сервісів у реальному часі.

Переваги використання Spring Boot

Spring Boot демонструє себе як оптимальне рішення для розробки мікросервісів завдяки наступним перевагам:

- Зниження витрат на розробку завдяки автоматизації конфігурації.

- Масштабованість: інтеграція з Kubernetes і Docker дозволяє запускати мікросервіси у хмарному середовищі.

- Екосистема: широка підтримка бібліотек і фреймворків для вирішення спеціалізованих задач (від Big Data до Machine Learning).

Висновки

1. Spring Boot є потужним інструментом для розробки мікросервісних додатків у середовищі Java. Завдяки своїм можливостям, таким як автоматичне налаштування, інтеграція з Spring Cloud і підтримка сучасних шаблонів архітектури, він значно спрощує розробку складних і масштабованих систем.

2. Успішне впровадження таких рішень залежить від правильного підходу до вирішення проблем масштабованості, безпеки та моніторингу.

3. Подальші дослідження у цій галузі можуть бути спрямовані на адаптацію нових можливостей Java (наприклад, Project Loom) для покращення продуктивності мікросервісів.

Література:

1. Льюїс Дж., Фаулер М. Мікросервіси: визначення нового архітектурного терміна / пер. з англ. А. Смирнова. URL: <https://martinfowler.com/articles/microservices.html>

2. Еванс Е. Проєктування, орієнтоване на домен: як впоратися зі складністю в серці ПЗ. Київ : Addison-Wesley, 2003.

3. Pivotal Software, Inc. Spring Boot Reference Documentation / пер. з англ. О. Автorenко. URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/>

4. Spring Cloud Team. Spring Cloud Documentation. URL: <https://spring.io/projects/spring-cloud>

5. Річардсон К. Шаблони мікросервісів: з прикладами на Java / пер. з англ. О. Мокровольського. Київ : Manning Publications, 2018.