УДК 004.7

Д. В. Кисюк
Р. О. Лашко

ПРИНЦИПИ АРХІТЕКТУРИ ТА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ВИСОКОНАВАНТАЖЕНИХ ВЕБ-СИСТЕМ

Вінницький національний технічний університет

Анотація

У статті систематизовано принципи архітектури та програмної реалізації сучасних веб-систем, орієнтованих на обробку високих навантажень. Детально проаналізовано вибір мов програмування Java та C++ як інструментів для реалізації ключових компонентів: корпоративних мікросервісів та високопродуктивних ядер систем відповідно. Розглянуто практичні аспекти реалізації архітектурних патернів стійкості, поліглотного зберігання даних та подієвої комунікації через написання конкретного програмного коду. Встановлено зв'язок між архітектурними рішеннями та їх безпосередньою реалізацією в програмних проектах.

Ключові слова: веб-архітектура, програмна реалізація, Java, C++, мікросервіси, високе навантаження, системне програмування.

Annotation

The article systematizes the principles of architecture and software implementation of modern web systems oriented toward high-load processing. It provides a detailed analysis of choosing Java and C++ programming languages as tools for implementing key components: corporate microservices and high-performance system cores, respectively. Practical aspects of implementing architectural patterns of resilience, polyglot data storage, and event-driven communication are examined through the development of specific program code. The relationship between architectural decisions and their direct implementation in software projects is established.

Keywords: *web architecture, software implementation, Java, C++, microservices, high load, systems programming.*

Вступ

Сучасна веб-розробка перетворилася зі створення окремих сторінок на проєктування складних розподілених систем, де програмний код є основним засобом втілення архітектурних ідей. Для студентів комп'ютерної інженерії, зокрема спеціальності системного програмування, критично важливим є розуміння не лише абстрактних принципів, але й того, як ці принципи реалізуються шляхом написання коду конкретними мовами програмування, такими як Java та C++ займають чіткі ніші в цьому процесі: перша є стандартом для побудови надійних, легко підтримуваних корпоративних сервісів, тоді як друга залишається незамінною для компонентів, де вимоги до продуктивності та контролю над ресурсами є абсолютними [3] [4]. Мета даної тези – чітко окреслити, як архітектурні вимоги до масштабованості, відмовостійкості та ефективності впливають на вибір мови програмування та формують підхід до організації програмного коду, який створює розробник.

Результат розробки

Сучасна веб-система реалізується як набір програмних компонентів, що взаємодіють між собою. Вибір мови програмування є першочерговим архітектурним рішенням, що визначає подальший процес написання коду.

Java - це мова реалізації мікросервісних екосистем. Архітектурний принцип розподілу на дрібні сервіси знаходить свою реалізацію у написанні кодових баз на Java з використанням фреймворка Spring Boot [1] [6]. Кожен мікросервіс – це окремий проект, код якого включає:

1. Класи-контролери (ановані @RestController), які програмно обробляють HTTP-запити [6]
2. Сервісні шари з бізнес-логікою.
3. Репозиторії для доступу до даних.
4. Конфігураційний код для реєстрації в системі оркестрації (наприклад, Eureka) та обробки помилок.

Програміст пише код, що реалізує патерни стійкості: наприклад, використання бібліотеки Resilience4j для програмної реалізації Circuit Breaker або Retry у методах сервісу [1].

C++ - це мова реалізації високонавантажених ядер. Архітектурна вимога до максимальної продуктивності та мінімальних затримок реалізується через написання низькорівневого коду на C++. Це включає:

1. Програмну реалізацію мережевих серверів з використанням бібліотек типу Boost.Asio для асинхронної обробки тисяч одночасних підключень.
2. Написання оптимізованих алгоритмів обробки даних (наприклад, фінансових транзакцій, алгоритмічних розрахунків) з ручним управлінням пам’яттю для уникнення накладних витрат [3].
3. Створення власних компонентів кешування або обробки потоків даних, де контроль над кожним циклом процесора є критичним.

Реалізація принципів архітектури через API та протоколи. Принцип слабкого зв’язку реалізується програмно: розробник пише код, що визначає контракти API (наприклад, використовуючи OpenAPI Specification для REST або файли .proto для gRPC). Код на Java генерує клієнтські та серверні stubs, а код на C++ – відповідні заголовки та класи, забезпечуючи типобезпечну комунікацію [2].

Поняття логічного сховища даних на рівні коду. Архітектурний вибір різних СУБД прямим чином впливає на код доступу до даних. Програміст пише:

1. JPA-запити або SQL-мапери (MyBatis) в коді Java для роботи з реляційними базами.
2. Код використання MongoDB Driver для роботи з документними колекціями.
3. Нативні C++ драйвери для таких баз даних, як Redis або Cassandra, коли потрібна максимальна швидкість у критичному шляху виконання [5].

Висновки

Таким чином, архітектура складних веб-систем безпосередньо диктує зміст програмного коду, який створюють розробники. Java виступає основним інструментом для програмної реалізації архітектури мікросервісів, забезпечуючи продуктивність командної розробки, надійність та екосистему готових рішень [6]. C++ залишається незамінним для програмної реалізації високонавантажених компонентів, де архітектурні вимоги до швидкодії та ефективності вимагають низькорівневого контролю. Розуміння цього взаємозв’язку між архітектурним задумом та практичним написанням коду є ключовим для фахівця в галузі комп’ютерної інженерії та системного програмування, дозволяючи обґрунтовано обирати інструменти та ефективно втілювати складні системні рішення.

СПИСОК ЛІТЕРАТУРИ

1. Newman, S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. O'Reilly Media, 2021.

2. Richards, M., Ford, N. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020.
3. Stroustrup, B. The C++ Programming Language. 4th ed. Addison-Wesley Professional, 2013.
4. Sharma, R. Oracle Certified Professional Java SE 17 Developer Study Guide. McGraw-Hill, 2022.
5. Kleppmann, M. Designing Data-Intensive Applications. O'Reilly Media, 2017.
6. Офіційна документація Spring Framework. https://spring.io/projects/spring-framework

Кисюк Дмитро Васильович — старший викладач кафедри обчислювальної техніки, Вінницький національний технічний університет, вул. Хмельницьке шосе 95, м. Вінниця, Україна, kneimad@gmail.com

Лашко Роман Олександрович — студент групи 1СП-25б, факультету інформаційних технологій та комп'ютерної інженерії, кафедри обчислювальної техніки, Вінницький національний технічний університет, вул. Хмельницьке шосе 95, м. Вінниця, Україна, romanlasko450@gmail.com

Kysiuk Dmytro V. — Senior Lecturer at the Department of Computer Engineering, Vinnytsia National Technical University, 95 Khmelnytske Shose St., Vinnytsia, Ukraine, kneimad@gmail.com

Lashko Roman O. — Student of the Department of Computer Engineering, 1SP-25b group, Vinnytsia National Technical University, 95 Khmelnytske Shose St., Vinnytsia, Ukraine, romanlasko450@gmail.com