УДК 621.395

ПЕРЕВАГИ ТА НЕДОЛІКИ ВИКОРИСТАННЯ МІКРОСЕРВІСНОГО ПАТТЕРНУ ПРОЕКТУВАННЯ

Маньківський В.Б., Коршак О.О.
Навчально-науковий інститут телекомунікаційних систем
КПІ ім. Ігоря Сікорського, Україна
E-mail: v.b.mankivskiy@gmail.com , korshak1234@gmail.com

ADVANTAGES AND DISADVANTAGES OF USING THE MICROSERVICE DESIGN PATTERN

The paper discusses the rationale and benefits of microservice architecture, including modularity, flexibility, scalability, and error isolation. Advantages such as rapid development, easy technology replacement, and speed of innovation are described. The disadvantages, such as system complexity, distributedness and testing issues, are highlighted.

У роботі розглянуто обґрунтування та переваги мікросервісної архітектури, включаючи модульність, гнучкість, масштабованість та ізоляцію помилок. Описано переваги, такі як швидка розробка, легка заміна технологій та швидкість інновацій. Висвітлені недоліки, як складність системи, проблеми розподіленості та тестування.

Обгрунтування вибору мікросервісної архітектури. Модульність і розділення функціональності: Мікросервісна архітектура дозволяє розділити великі програми на невеликі, незалежні сервіси. Це полегшує розробку, тестування і підтримку системи, оскільки кожен сервіс може бути розроблений, тестований і оновлюваний окремо.

Гнучкість і масштабованість: Мікросервісна архітектура дозволяє гнучко масштабувати окремі сервіси залежно від їхньої навантаженості та вимог. Це дозволяє оптимізувати використання обчислювальних ресурсів та забезпечує більшу ефективність системи.

Ізоляція помилок: У мікросервісній архітектурі помилки у одному сервісі не впливають на решту системи, оскільки кожен сервіс працює незалежно. Це підвищує надійність та стійкість системи до збоїв.

Прискорена розробка та розгортання: Розробка окремих мікросервісів може проводитися паралельно різними командами, що сприяє швидкій розробці та впровадженню нового функціоналу.

Легка заміна технологій: Вибір мікросервісної архітектури робить систему менш залежною від конкретних технологій. Кожен сервіс може використовувати технології, що найкраще підходять для його завдань.

Підтримка розподіленого розгортання: Мікросервіси добре підходять для розподіленого розгортання на різних серверах або хмарних.

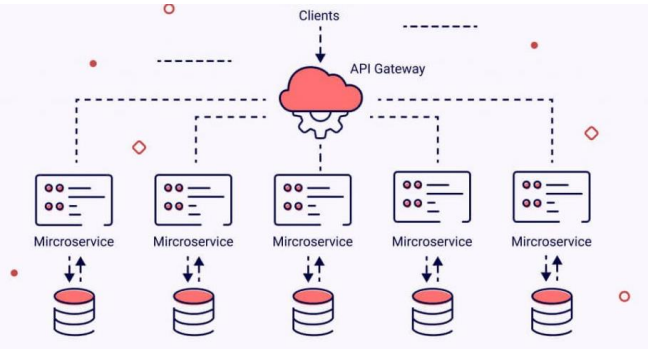

Рисунок 1. Microservice architecture.

На Рис.1 показаниа типова мікросервісна архітектура з окрменим виділенням БД для кожного сервісу [1]. В даній архітектурі використаний паттерн API-gateway, який розподіляє всі запити до відповідного сервісу. При цьому, явно виділяється недолік такого підходу в тому, що є одна точки виоду із строю, а також централізований підхід до керування.

Окрмі того, API-gateway вионує додаткові функції: логування, трасування, аутентифікацію. Дані додаткові функції підвищують гнучкість такої системи.

Переваги та недоліки мікросервісної архітектури. Мікросервісна архітектура визначається сукупністю властивостей, які сприяють ефективному масштабуванню застосунків як у горизонтальному, так і у вертикальному напрямках. Крім того, ці властивості сприяють створенню надійних та відмовостійких застосунків [2].

Масштабованість: Однією з ключових переваг мікросервісної архітектури є її здатність ефективно масштабуватися як горизонтально, так і вертикально. Це робить процес розробки та розгортання економічно більш вигідним і менш часозатратним, особливо у порівнянні з монолітними архітектурами. У монолітних системах зазвичай єдиним методом масштабування є копіювання всієї програми, навіть якщо потрібно збільшити обсяг лише певних модулів [3]. Це стає особливо актуальним, коли збільшується кількість користувачів застосунку.

Відмовостійкість: У мікросервісних застосунках, завдяки їхній низькій зв'язності, виявляється природна властивість відмовостійкості. Це означає, що у випадку, якщо певний модуль системи припиняє свою роботу, його можна легко вимкнути, замінити або відлагодити, не призводячи до зупинки роботи всієї системи в цілому.

Технологічність: Кожен індивідуальний мікросервіс у системі може мати свій власний технологічний стек. Крім того, цей стек можна легко оновлювати. Неможливо враховувати всі аспекти від кожного мікросервісу, можливості все ще значно ширші порівняно з монолітним застосунком.

Простота у розумінні окремого мікросервісу: Кожен окремий мікросервіс є меншим за розміром, ніж повноцінний монолітний застосунок. Одночасно код в межах мікросервісу має високий рівень пов’язаності, і так, оскільки система має низьку зв'язність, кодова база є набагато простішою.

Незважаючи на велику кількість сильних сторін, мікросервісні застосунки також мають свої слабкі сторони [4]. Деякі з цих недоліків не є

проблемою в монолітних застосунках, що підкреслює важливість правильного використання та реалізації для кожної:

- Складність системи в цілому: Кожен окремий мікросервіс може бути досить зрозумілим, проте при розгляданні системи в цілому вся мікросервісна архітектура стає досить складною та значною важчою для розуміння, ніж монолітна.

- Розподіленість: Мікросервісна архітектура побудована на ідеї незалежності мікросервісів. Основною складністю є саме взаємодія цих мікросервісів та використання досить складних протоколів спілкування.

- Відлагодження: З урахуванням вищезгаданих проблем можна додати, що відлагодження системи в цілому є складнішою задачею.

Висновок. Мікросервісна архітектура є потужним інструментом для розробки сучасних програмних систем. Обгрунтовано вибір цієї архітектури з точки зору модульності, гнучкості, масштабованості, ізоляції помилок, швидкості розробки та розгортання, легкої заміни технологій, швидкості інновацій та підтримки розподіленого розгортання [5]. Вона сприяє підвищенню ефективності, надійності і стійкості програмних систем, а також полегшує їхнє розвиток і супровід. Проте, разом з численними перевагами, мікросервісна архітектура має і свої недоліки. Складність системи, проблеми розподіленості, складнощі у тестуванні та відлагодженні можуть стати серйозними викликами для розробників. Однак правильне планування, проектування і впровадження можуть допомогти подолати ці проблеми і максимізувати переваги .

Література

1. Приходченко, С. Д., Роина, К. С., Поштак, Р. В. (2018). Обгрунтування вибору мікросервісної архітектури в порівнянні з монолітною. [Електронний ресурс]. URL: http://ir.nmu.org.ua/handle/123456789/153837
2. Новак, В. В. (2022). Порівняння монолітної та мікросервісної архітектур на прикладі корпоративного застосунку. [Електронний ресурс]. URL: https://dspace.znu.edu.ua/jspui/handle/12345/9531
3. Степанова, Н. І. (2022). АНАЛІЗ МІКРОСЕРВІСНИХ ТА МОНОЛІТНИХ АРХІТЕКТУР. Міжнародний науковий комітет, 91. [Електронний ресурс]. URL: https://www.dnu.dp.ua/docs/ndc/2022/materiali/25_%D0%9C%D0%9F%D0%97%D0%86%D0%A1-2022-1.pdf#page=91
4. Newman, S. (2019). Monolith to microservices: evolutionary patterns to transform your monolith. O'Reilly Media. [Електронний ресурс]. URL: https://books.google.com.ua/books?hl=uk&lr=&id=nNa_DwAAQBAJ&oi=fnd&pg=PP1&dq=Monolith+to+microservices:+evolutionary+patterns+to+transform+your+monolith.+O%27Reilly+Media&ots=eiUW1wnefH&sig=DNeWIzndCLJrTox2-7RSz6OgYj4&redir_esc=y#v=onepage&q=Monolith%20to%20microservices%3A%20evolutionary%20patterns%20to%20transform%20your%20monolith.%20O'Reilly%20Media&f=false
5. Thönes, J. (2015). Microservices. IEEE software, 32(1), 116-116. [Електронний ресурс]. URL: https://ieeexplore.ieee.org/abstract/document/7030212