УДК 378:004.384

ОБҐРУНТУВАННЯ ВИБОРУ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ В ПОРІВНЯННІ З МОНОЛИТНОЮ

С.Д. Приходченко, К.С. Родна, Р.В. Поштак
(Україна, Дніпро, НТУ «Дніпровська політехніка»)

Ріст будь якого веб-ресурсу означає збільшення кількості користувачів, а це призводить до збільшення кількості запитів, які цей веб-ресурс повинен мати змогу опрацювати щосекунди. Для подолання даної проблеми, зазвичай, використовують масштабування.

Використання мікросервісної архітектури є одним з часто використовуваних методів горизонтального масштабування на даний момент. Мікросервісна архітектура має доволі значний недолік – збільшення значення затримок при комунікації між сервісами. Тобто, при ідентичній реалізації сервісів, монолітна архітектура може мати кращі показники швидкодії, ніж мікросервісна. З іншого боку, мікросервісна архітектура дає більше можливостей для масштабування, що є суттєвою перевагою для систем, що швидко розвиваються.

Таким чином, як мікросервісна, так і монолітна архітектури мають як позитивні, так і негативні властивості, тому визначення архітектури, яка буде використана при побудові веб-сервісів, дуже важливе і має безпосередній вплив на те, як система буде розвиватись надалі.

Аналіз існуючих джерел інформації показує, що перед використанням мікросервісної архітектури необхідно провести аналіз розроблюваної системи і визначити, наскільки прийнятними є показники затримки при комунікації між веб-сервісами, і чи не призведуть вони до низької швидкодії системи і неможливості подальшого її масштабування. Таким чином, важливо оцінити обґрунтованість використання мікросервісної архітектури для побудови сервісів конкретного веб-ресурсу.

Монолітна архітектура передбачає реалізацію всіх сервісів ресурсу як єдиної програмної системи. Тобто всі сервіси реалізовані за допомогою одного набору технологій (і мови програмування) і використовують загальні бібліотеки коду. Всі сервіси працюють з одним сервером баз даних, що дозволяє кожному сервісу звертатися до бази даних безпосередньо (рис. 1).

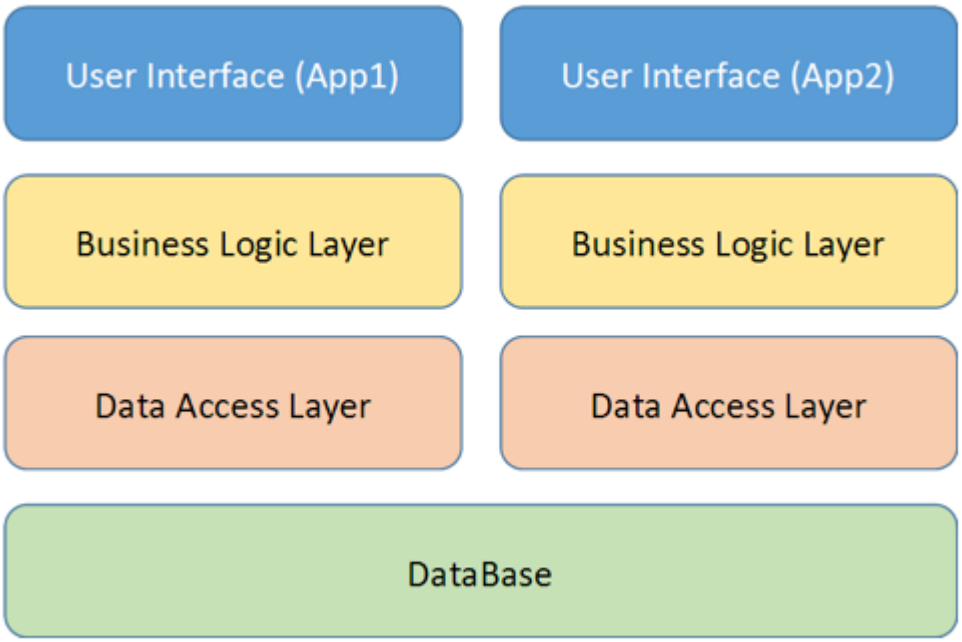

Рис. 1. Програмна реалізація монолітної архітектури

Мікросервісна архітектура використовує інший підхід. Кожен сервіс реалізований як окрема програмна система, часто у кожного сервісу є своя база даних. Оскільки сервіси не мають доступу до бази даних іншого сервісу – доступ до таких даних здійснюється викликом інших сервісів ресурсу. Часто сервіси групують, якщо вони реалізують схожий, або тісно пов'язаний функціонал. Прикладом такої ситуації може бути процес реєстрації і аутентифікації. Так, за ці два процеси будуть відповідати кілька сервісів, але оскільки вони будуть працювати з сутністю «Користувач», має сенс використовувати загальну базу даних. Приклад реалізації мікросервісного підходу надано на рисунку 2.

Залежно від значень характеристик, описаних вище, можна визначити яка архітектура найкраще підходить до кожного конкретного проекту. Для оцінки буде використана система рейтингу. В таблиці 1 відображений рейтинг значень характеристик. Опис значень, які можуть отримати монолітна та мікросервісна архітектури в залежності від значень характеристик:

– -1 – вибір архітектури негативно вплине на розвиток проекту;

– 0 – вибір архітектури не буде впливати на подальшу розробку проекту;

– +1 – вибір архітектури позитивно вплине на розвиток проекту;

– +2 –вибір архітектури значним чином позитивно вплине на розвиток проекту.

Використання даної шкали оцінювання обумовлений тим, що, таким чином, недоліки по одній із характеристик зможуть компенсуватись позитивним впливом по іншій.

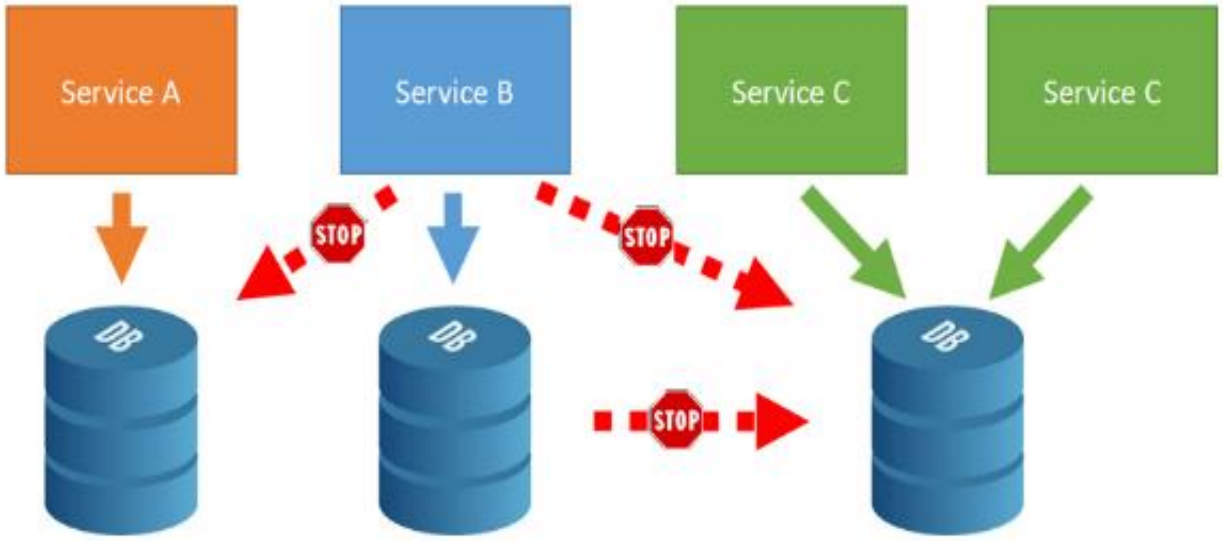

Рис. 2. Програмна реалізація мікросервісної архітектури

Значення рейтингу обумовлені наступними факторами:

1) вимоги до швидкодії – при відсутності вимог до швидкодії, висока складність розробки мікросервісів не обґрунтована. З іншого боку, оскільки у мікросервісів кращі показники масштабування, можна отримати суттєвий виграш при їх використанні, де швидкодія дійсно важлива;

2) при використанні мікросервісів з маленькою командою втрачається виграш в розділенні відповідальності між сервісами, а при використанні монолітів з великою командою – кожен член команди має дуже мало інформації про систему, за яку він відповідальний;

3) мікросервіси показують себе краще при частому встановленні нової версії, оскільки воно здійснюється швидко і не блокує решту сервісів;

4) при різносторонній технологічній орієнтованості команди, мікросервіси є кращими, оскільки кожен член команди має можливість використовувати ті технології, в яких у нього найбільше знань та досвіду;

5) очевидно, що відсутність досвіду в використанні мікросервісів негативно вплине на процес розроблення. Підсумовуючи результати аналізу (табл. 1), слід визначити, що мікросервісна архітектура має суттєво вищий рейтинг, якщо скласти значення за усіма показниками. Цей результат може бути уточнений, якщо використати вагові коефіцієнти для окремих показників.

Таблиця 1

Рейтинг значень характеристик проекту та його впливу на вибір архітектури

№	Назва характеристики	Значення характеристики	Монолітна архітектура	Мікросервісна архітектура
1	Вимоги до швидкодії	Відсутні	+1	0
		Не критично	0	0
		Критично	0	+2
2	Розмір команди	Маленька	+1	-1
		Середня	0	0
		Велика	-1	+2
3	Частота встановлення нових версій	Відразу після розробки	-1	+1
		По графіку	0	0
4	Технологічна орієнтованість команди	Вузька	+1	0
		Різностороння	-1	+1
5	Досвід роботи з мікросервісами	Є	0	+1
		Немає	0	-1

Мікросервіси є сучасною концепцією реалізації сервісів для систем, що розвиваються. Даний підхід використовується такими ресурсами як Amazon і Netflix, що, безсумнівно, є підтвердженням актуальності підходу. Важливо, так само, розуміти, що використання даного підходу має бути виправдано, так як є характеристики, в яких монолітна архітектура показує себе значно краще. В результаті порівняння були визначені характеристики, які повинні впливати на процес вибору веб-сервісної архітектури, а також запропоновано просту методику для вибору архітектури залежно від цих характеристик.

ПЕРЕЛІК ПОСИЛАНЬ:

1. Newman, Sam Building Microservices [Text] / Sam Newman. –O'Reilly Media, 2015. – P. 53-58.

2. Richardson,Chris Microservice Patterns [Text] / Chris Richardson. – Fall, 2017. – 375 p.

3. Микросервисы: пожалуйста, не нужно [Электронный ресурс] // Хабрахабр. – Режим дос- тупа:https://habrahabr.ru/post/311208/. – 24.02.2017.

4. Микросервисы (Microservices) [Электрон- ный ресурс] // Хабрахабр. – Режим досту- па:https://habrahabr.ru/post/249183/. – 24.02.2017.

5. Сначала – монолит, или правильный путь к микросервисной архитектуре [Электронный ресурс] // TProger. – Режим доступа:https://tproger.ru/ translations/monolithfirst/.– 12.02.2017.

6. Преимущества и недостатки микросервис- ной архитектуры [Электронный ресурс] // Записки программиста – Режим доступа:http://eax.me/ micro-service-architecture/. – 14.02.2017.

7. Microservices [Electronic resource] // Wikipedia. The free encyclopedia. – Access mode: https://en.wikipedia.org/wiki/Microservices. – 20.01.2017.

8. What is Microservices Architecture? [Electronic resource] // Smartbear. – Access mode: https://smartbear.com/learn/api-design/what-aremicroservices/. – 22.01.2017.