УДК 004.415.2

Р. Г. КЛАПЧУК, В. С. ХАРЧЕНКО

Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ», Україна

МОНОЛІТНІ ВЕБ-СЕРВІСИ ТА МІКРОСЕРВІСИ: ПОРІВНЯННЯ ТА ВИБІР

Метою статті є аналіз двох найбільш використовуваних архітектурних підходів до побудови веб-сервісів – монолітним і мікросервісним. Актуальність теми обумовлено постійним зростанням кількості користувачів веб-ресурсів, що, в свою чергу, викликає необхідність безперервного масштабування. Дано визначення основних понять, пов'язаних з предметною областю, селектовано і проаналізовано основні характеристики веб-сервісів. Проведено порівняння монолітної та мікросервісної архітектур. Надано рекомендації щодо вибору архітектури для конкретного веб-ресурсу.

***Ключові слова**: веб-сервіси, монолітні сервіси, мікросервіси, сервіс-орієнтована архітектура.*

1. Аналіз проблеми та постановка завдання

1.1. Актуальність та мотивація

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[1].

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 [1, 2] 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.

1.2. Огляд літератури

Проблема вибору архітектури веб-сервісів є часто обговорюваною в середі розробників програмного забезпечення та системних архітекторів. Різні джерела по різному оцінюють ситуацію, приводячи як позитивні, так і негативні сторони використання мікросервісної архітектури і відмови від монолітної архітектури.

Аргументами джерел, які відносяться до використання мікросервісів негативно [3], найчастіше є вищі значення затримок при міжсервісній комунікації та більш висока складність експлуатації. Джерела ж, які, навпаки, вказують на доцільність використання мікросервісів [4], приводять такі аргументи, як гнучкість процесів масштабування та встановлення веб-сервісів, спрощення розроблення для великих команд шляхом більш чіткого розмежування відповідальності.

Більшість же інших ресурсів [5, 6, 7] описує доцільність використання мікросервісної архітектури як залежне від конкретної ситуації. Нажаль, конкретних вказівок, які можна використати при виборі архітектури знайдено не було.

1.3. Мета і структура

Мета статті – дослідження сучасних архітектур веб-сервісів, їх відмінностей та оцінка доцільності використання кожної. Аналіз і порівняння архітектур виконано за наступними характеристиками:

1. Швидкодія – характеристика, яка описує базові показники швидкості, з якою користувач веб-ресурсу може отримати доступ до запитаних даних.
2. Масштабованість – характеристика, яка

описую наскільки просто та швидко система масштабується, тобто адаптується до збільшення навантаження.

3. Складність розробки – характеристика, яка описує час, який необхідний команді на розробку системи, а також необхідний рівень спеціалістів, які над нею працюють.

4. Складність встановлення – характеристика, яка описує час та вартість встановлення нової версії продукту для використання кінцевими користувачами.

Структура статті наступна: у другому розділі проведено порівняльний аналіз монолітної та мікросервісної архітектур. В третьому розділі представлено характеристики, які впливають на вибір архітектури побудування веб-сервісів та описано як вибрати архітектуру веб-сервісів в залежності від характеристик конкретного веб-ресурсу.Також підведено підсумки і визначено напрями подальших досліджень.

2. Порівняння монолітної та мікросервісної архітектури

2.1. Програмна реалізація

Одною з найбільш явних відмінностей є відмінність в саме програмній реалізації систем. Саме відмінність у реалізації систем, побудованих за монолітною та мікросервісною архітектурою, впливає на решту характеристик.

2.1.1. Монолітна архітектура

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).

2.1.2. Мікросервісна архітектура

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 автентифікації. 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.

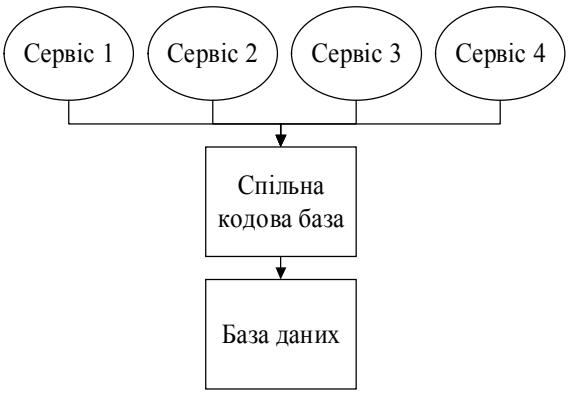

Рис. 1. Програмна реалізація монолітної архітектури

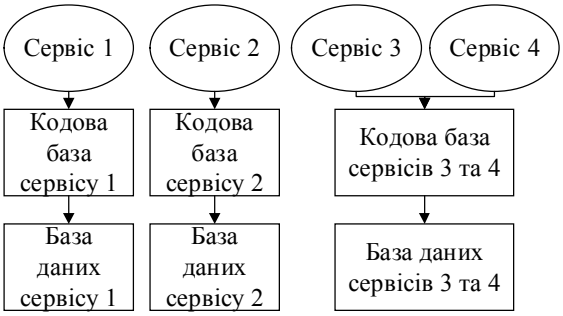

Рис. 2. Програмна реалізація мікросервісної архітектури

2.2. Доступ до загального коду та даних

Як і було описано раніше, монолітна архітектура реалізується єдиною програмною системою, яка працює з єдиною базою даних і кодовою базою. Таким чином, кожен сервіс має доступ до всіх ресурсів додатку.

При реалізації мікросервісної архітектури, кожен сервіс має свою кодову базу та базу даних і тільки він має до них доступ. Таким чином, мікросервісній архітектурі притаманний більш складний доступ до ресурсів, які відносяться до інших сервісів[8].

2.3. Розгортання нової версії

Важливою характеристикою при оцінці процесу розроблення будь-якого ресурсу є Time to market. Ця характеристика оцінює наскільки швидко розроблена функціональність може бути використана кінцевим користувачем.

При використанні монолітної архітектури, встановлення нової версії навіть одного сервісу

вимагає переустановлення всієї програмної системи. Так як все більше і більше ресурсів розгортаються в «хмарі» – постійне переустановлення всієї програми дуже витратне. Крім того, встановлення всього продукту займає немало часу, тому ресурс стає недоступним на час встановлення, що, в свою чергу, може бути недопустимим для деяких веб-ресурсів.

Мікросервісна архітектура дозволяє перевстановлювати тільки ті сервіси, в яких були зміни [9, 10]. Це дозволяє зменшити час розгортання, а також дозволяє не блокувати інші сервіси в процесі. Крім того, при встановленні лише конкретних сервісів простіше контролювати стабільність ресурсу (при виникненні помилок або несподіваної зміни поведінки простіше виявити джерело)[11].

2.4. Масштабування

Масштабованість описує здатність системи справлятися зі зростаючими навантаженнями. Існує два основних типи масштабування – вертикальне і горизонтальне.

Вертикальне масштабування передбачає збільшення потужності існуючих ресурсів системи. Приклад –встановлення більш потужного процесора на сервер.

Горизонтальне масштабування більш складне, тому що найчастіше вимагає зміни в коді системи. Горизонтальне масштабування – процес розбиття системи на більш дрібні частини і розміщення їх на окремих фізичних вузлах. Найпростішим і часто використовуваним варіантом горизонтального масштабування є використання розподілених систем, які дозволяють розподіляти обробку запитів на кілька фізичних вузлів.

Саме в горизонтальному масштабуванні є одна важлива перевага мікросервісів. Так як кожен сервіс є окремою програмою, при підвищенні навантаження на один із сервісів його можна просто встановити на окремий фізичний вузол. Таким чином, завантаження цього сервісу не вплине на стабільність інших сервісів, а також дозволить вертикально масштабувати цей сервіс в залежності від того, які ресурси він споживає найбільше.

2.5. Швидкодія

На жаль, мікросервісна архітектура має один дуже важливий недолік – без масштабування, системи, побудовані за нею, мають менші показники продуктивності ніж ті, які реалізовано за допомогою монолітної архітектури. Цей факт викликаний тим, що неможливість отримувати доступ до даних безпосередньо викликає додаткові мережеві затримки при взаємодії сервісів.

Використання монолітної архітектури дозволяє використовувати загальну логіку безпосередньо через виклик коду, який знаходиться в загальних бібліотеках коду. Те ж саме стосується і доступу до даних (рисунок 3).

При використанні мікросервісної архітектури ж необхідне створення додаткового сервісу, який і буде надавати доступ до функціональності, необхідної обом сервісам. Для доступу до загальної логіки кожен сервіс має сформувати запит відповідно до обраного протоколу, відправити його і дочекатися відповіді (рисунок 4).

Необхідність виклику проміжних сервісів створює додаткову затримку за рахунок необхідності формування запиту/відклику та додаткової передачі їх через мережу.

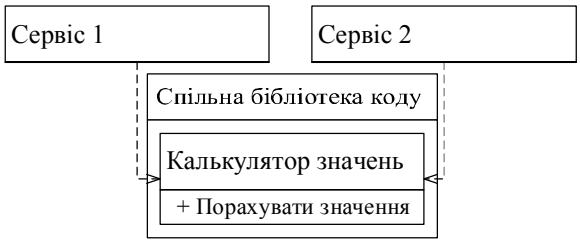

Рис. 3. Доступ до коду в монолітній архітектурі

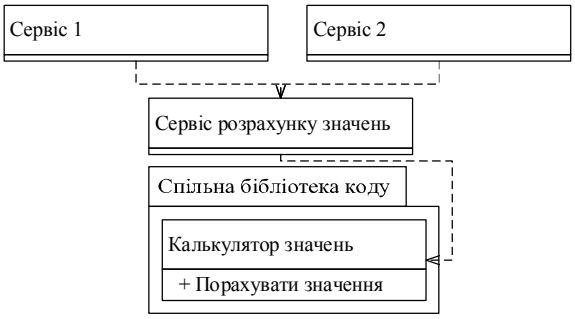

Рис. 4. Доступ до коду в мікросервісній архітектурі

3. Вибір веб-сервісної архітектури

3.1. Вибір характеристик

Вибір архітектури для нового проекту, а тим більше, перехід існуючого проекту на нову архітектуру має бути аргументований. Неправильно прийняте рішення коштуватиме дуже дорого, оскільки може призвести до нестабільності системи та неможливості роботи при збільшенні навантаження.

Для вибору архітектури проект має бути проаналізований за наступними властивостями:

1. Вимоги до швидкодії. Можливими значеннями даної характеристики в контексті даної статті є: відсутні (швидкодія не важлива), не критично (швидкодія має значення, але для складних завдань

високий час очікування прийнятний), критично (повинна бути реалізована максимальна можлива швидкодія (як, наприклад, в системах реального часу)).

2. Розмір команди, що працює над продуктом. Тут мається на увазі кількість людей, які одночасно працюють над продуктом. В контексті даної статті за основу будуть взяті такі розміри команд: 1-2 розробника – маленька команда, 3-9 – середня команда, 10 і більше – велика команда.

3. Очікувана частота встановлення нових версій. В контексті даної статті за основу будуть взяті такі значення характеристики: відразу після розробки нового функціоналу, по графіку (щоночі, раз в місяць і так далі).

4. Технологічна орієнтованість команди. Тут мається на увазі той факт, чи всі члени команди підготовлені та показують максимальну швидкість та ефективність розробки на одному і тому ж напрямку технологій.

5. Наявність у команди досвіду роботи з мікросервісами.

3.2. Вибір архітектури

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.

Значення рейтингу обумовлені наступними факторами:

1) вимоги до швидкодії – при відсутності вимог до швидкодії, висока складність розробки мікросервісів не обгрунтована. 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

Висновки

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.

Наступним кроком досліджень є більш глибоке дослідження цих альтернативних архітектур з використанням математичних моделей систем масового обслуговування або імітаційного моделювання. Перспективним, на наш погляд, є використання адаптивних архітектур, які змінюються залежно від характеристик потоків задач, а також з урахуванням факторів надійності і безпеки компонентів сервісів.

Література

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 доступа:https://habrahabr.ru/post/311208/. – 24.02.2017.

4. Микросервисы (Microservices) [Электронный ресурс] // Хабрахабр. – Режим доступа: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 микросервисной 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-are-microservices/. – 22.01.2017.

9. Richardson, C. Pattern: Microservice Architecture [Electronic resource] / C. Richardson // Kong. – Access mode: http://microservices.io/patterns/microservices.html. – 22.01.2017.

10. Microservices [Electronic resource] // M. Fowler. – Access mode:https://martinfowler.com/articles/microservices.html. – 17.02.2017.

11. What are microservices? [Electronic resource] // The open source way. – Access mode:https://opensource.com/resources/what-are-microservices. – 20.02.2017.

References

1. Newman,Sam. *Building Microservices*. O'Reilly Media, 2015,pp. 53-58.

2. Richardson,Chris. *Microservice Patterns*. Fall Publ., 2017. 375 p.

3. *Mikroservisy: pozhalujsta, ne nuzhno* [Microservices: please, don't].Habrahabr. Available at:https://habrahabr.ru/post/311208/(accessed 24.02.2017) (In Russian).

4. *Mikroservisy (Microservices)*[Microservices (Microservices)].Habrahabr.Available at:https://habrahabr.ru/post/249183/(accessed 24.02.2017) (In Russian).

5. *Snachala – monolit, ili pravil'nyj put' k mikroservisnoj arhitekture* [Start with monolith, or the right way to microservice architecture].*TProger*.Available at:https://tproger.ru/translations/ monolithfirst/(accessed 12.02.2017) (In Russian).

6. *Preimushhestva i nedostatki mikroservisnoj arhitektury* [Advantages and disadvantages of microservice architecture].Zapiski programmista.Available at: http://eax.me/micro-service-architecture/ (accessed 14.02.2017) (In Russian).

7. *Microservices.*Wikipedia. The free encyclopedia. Available at: https://en.wikipedia.org/wiki/Microservices (accessed 20.01.2017).

8. *What is Microservices Architecture?*Smartbear.Available at: https://smartbear.com/learn/api-design/ what-are-microservices/(accessed 22.01.2017).

9. Richardson,Chris. *Pattern: Microservice Architecture*.Kong. Available at: http://microservices.io/patterns/microservices.html (accessed 22.01.2017).

10. *Microservices*. Martin Fowler.Available at: https://martinfowler.com/articles/microservices.html (accessed 17.02.2017).

11. *What are microservices?*The open source way. Available at: https://opensource.com/resources/what-are-microservices (accessed 20.02.2017).

Поступила в редакцию 5.02.2017, рассмотрена на редколлегии 16.02.2017

МОНОЛИТНЫЕ ВЕБ-СЕРВИСЫ И МИКРОСЕРВИСЫ: СРАВНЕНИЕ И ВЫБОР

Р. Г. Клапчук, В. С. Харченко

Целью статьи является анализ двух наиболее используемых архитектурных подходов к построению веб-сервисов – монолитным и микросервисным. Актуальность темы обусловлена постоянным ростом количества пользователей веб-ресурсов, что, в свою очередь, вызывает необходимость непрерывного масштабирования. Даны определения основных понятий, связанных с предметной областью, выделены и проанализированы основные характеристики веб-сервисов, а также проведено сравнение монолитной и микросервисной архитектур. Кроме того, статья содержит рекомендации по выбору архитектуры для конкретного веб-ресурса.

Ключевые слова: веб-сервисы, монолитные сервисы, микросервисы, сервис-ориентированная архитектура.

MONOLITH WEB-SERVICES AND MICROSERVICES: COMPARATION AND SELECTION

R. G. Klapchuk, V. S. Kharchenko

The paper is dedicated to analysis of two most commonly used architectures for creation of web services– monolith and microservice. Relevance of the subject is caused by the continuous growth of web resources users and the need of service scaling. The common notions related to the subject is defined, the main properties of web services are analyzed, monolith and microservice architectures are compared. Besides, the article contains recommendations regarding choice of the most suitable architecture for the very web resource.

Keywords: web-services, monolithic services, microservices, service-oriented architecture.

Клапчук Руслан Григорович – магістрант каф. комп'ютерних систем та мереж, Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ», Харків, Україна, e-mail: rkjorged@gmail.com.

Харченко В'ячеслав Сергійович – д-р техн. наук, проф., зав. каф. комп'ютерних систем та мереж, Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ», Харків, Україна, e-mail: v.kharchenko@csn.khai.edu.

Klapchuk Ruslan Grygorovych– graduate student of Dept. of Computer Systems and Networks, NationalAerospace University named after N. Ye. Zhukovsky "KhAI", Kharkov, Ukraine, e-mail: rkjorged@gmail.com.

KharchenkoVyacheslav Sergeyevich – Dr. of Technical Science, prof., head of Dep. computer systems and networks,National Aerospace University named after N. Ye. Zhukovsky "KhAI", Kharkov, Ukraine, e-mail: v.kharchenko@csn.khai.edu.