DOI:
УДК 004.051, 004.053, 004.057.4, 004.057.5
А.С. Білецький, аспірант, anton.beletsky@gmail.com
Л.М. Божуха, к.ф.-м.н., доцент, доцент кафедри МЗЕОМ, bozhukha.li@gmail.com
Дніпровський національний університет ім. Олеся Гончара, м. Дніпро

ПРО МЕТОДИ ГНУЧКОГО НАДАННЯ ТА МОДИФІКАЦІЇ РЕСУРСІВ У ІНТЕРФЕЙСАХ ВЕБ-СЛУЖБ У МІКРОСЕРВІСНІЙ АРХІТЕКТУРІ

Мікросервісна архітектура дозволяє розробляти розподілені системи використовуючи незалежні, які легко пов'язуються, і придатні до спільного використаннясервіси. Вони можуть бути реалізовані на різних технологіях і мовах програмування, тому потребують налагодження каналів зв'язку один із одним. Мікросервіс, у свою чергу, не має знати які саме клієнти будуть використовувати його програмний інтерфейс. Тому для зручності використання цей інтерфейс повинен буди якомога гнучкішим. Ця стаття спрямована на огляд методів побудови гнучких програмних інтерфейсів веб-служб у мікросервісній архітектурі.
***Ключові слова**: мікросервіси; програмний інтерфейс; клієнт; сервер; веб-ресурс; уніфікований локатор ресурсів.*

Microservice architecture allows you to develop distributed systems using independent, easily interconnected, and shared services. They can be implemented in different technologies and programming languages, so they need to establish communication channels with each other. The microservice, in turn, does not need to know which clients will use its software interface. Therefore, for ease of use, this interface should be as flexible as possible. This article aims to review methods for building flexible web service software interfaces in microservice architecture.
***Keywords**:microservices;application programming interface (API); client; server; web-resource; Uniform Resource Locator (URL).*

Постановка проблеми

Мікросервісна архітектура є архітектурним стилем з певним набором рекомендацій і правил. Для реалізації програмних інтерфейсів мікросервісів наразі немає єдиного стандарту, що змушує розробників бувати власні формати API та ускладнює інтеграцію з іншими вебсервісами.

Перед розробниками програмного забезпечення практично завжди постає задача виявлення методів побудови програмних інтерфейсів у мікросервісних системах та надання гнучких таким конструкціям гнучкості для подальшої можливої модифікації.

При проєктуванні системи актуальним є питання щодо обрання методів та підходів, які дозволяють модифікувати ресурси на рівні окремих мікросервісів і на рівні усієї системи в цілому. Дослідженням цього питання може стати аналіз існуючих методів та підходів гнучкого надання та модифікації ресурсів у інтерфейсах веб-служб.

Для вирішення завдання необхідно проведення порівняльного огляду існуючих методів. Зокрема, проведенню аналізу методів представлення ресурсів та модифікації ресурсів на рівні сервісу, методів побудови та масштабування програмних інтерфейсів, присвячена ця публікація.

Аналіз останніх досліджень та публікацій

Для розроблення інформаційних та комунікаційних систем необхідні визначені принципи організації, проєктування та розроблення систем. Для web-програмування актуальною залишається концепція мікросервісної архітектури.

Проблематикою концепції мікросервісної архітектури є нестандартизовані формати обміну повідомлень.

Прикладом є стандартизований протокол (SOAP), який призначений для обміну даними через визначений формат XML. Цей форматзменшує швидкість обміну великого об'єму даних між мікросервісами.

На сьогоднішній день можна виділити основні підходи до проєктування інтерфейсів.

Підхід REST є архітектурним стилем, який містить визначений набор обмежень для веб-сервісів. Наведений стиль не зберігає стан системи та використовує стандартизований підхід до створення веб API [1]. Відмінністю gRPC є договір в обміні повідомлень. Підхід REST визначає взаємодію за допомогою стандартизованих термінів у запитах, а підхід gRPC функціонує на ідеї відносин клієнт-сервер. При використанні стратегіїg RPCбільшу частину відповідальності надається клієнту. Цей підхід використовує вивантаження обробки даних тавідповідні обчислення на віддаленому сервері, що є доречним для рішень, які вимагають комунікацій на замовлення між користувачами для малопотужних пристроїв. Для стратегії клієнт-сервер обирають підхід GraphQL, який є більш архітектурою та договір додатково визначається самими ресурсами.

Формулювання мети дослідження

Метою дослідженняє виявлення методів побудови та надання гнучких програмних інтерфейсів у мікросервісних системах.

Для вирішення цієї задачі використовуються методи та підходи, які дозволяють будувати програмні інтерфейси для гнучкого надання і модифікації ресурсів, як на рівні окремих мікросервісів, так і на рівні усієї системи вцілому.

Задачею на даному етапі дослідження є формування стратегії обрання методів побудови та надання гнучких програмних інтерфейсів у мікросервісних системах при проєктуванні системи для подальшого розроблення технології модифікації ресурсів та масштабування системи.

Виклад основного матеріалу

При проведенні огляду методів гнучкого надання та модифікації ресурсів у інтерфейсах веб-служб можна виділити декілька підходів для обрання стратегії проєктування системи:методи роботи з ресурсами на рівні сервісу (методи побудови та методи представлення), методи побудови програмних інтерфейсів та методи масштабування програмних інтерфейсів.

При розгляді методів побудови гнучких програмних інтерфейсів існує безліч підходів, кожен з яких не є універсальним рішенням і завжди залишається необхідність виконання операцій вибірки, об’єднання та проєкції реляційної алгебри,модифікації та видалення ресурсів/міжзв’язків.

До гнучкого інтерфейсу між клієнтом та сервером вибудовують наступні умови: ідентифікація ресурсів, змінення ресурсів для оброблення через представлення та змінення стану системив залежності від дій клієнтів.

Спрощення процесу проєктування та розроблення мікросервісних систем є наслідком створення інтерфейсу мікросервісом.

Впровадження технології по наданню гнучкого інтерфейсу полегшує додавання нових незалежних модулів у мережу.

Умова ідентифікації ресурсіввизначена представленням інформації. Сучасний сервіс може бути представлений десктоп-клієнтом у вигляді браузеру, мобільним клієнтом у вигляді мобільного додатку, машинним клієнтом у вигляді веб-сервісу та інші. Кожен з цих клієнтів потребує різну інформацію та форму її представлення.

Методи побудови гнучкого інтерфейсу веб-служб залежать від розміру системи для якої проєктуються API. В цілому їх можна розділити:на методи на рівні сервісу, методи на рівні системи.

Перші можна застосовувати в монолітних системах, системах з сервіс-орієнтованою архітектурою та окремих мікросервісах. А другі використовуються для об’єднання API пов’язаних мікросервісів у єдиний високорівневий API.

При розгляді методів представлення ресурсів на рівні сервісудля розроблення такого гнучкого інтерфейсу можна використати реляційну теорію та адаптувати її принципи на інтерфейс веб-сервісу [2].

Найчастіше клієнти роблять запити для отримання колекції ресурсів, які можуть бути представлені в json або xml форматах. Незалежно від формату, кожен ресурс представляє собою об’єкт із набором атрибутів певного типу. Цей ресурс можна розглянути як реляційне відношення $R(A_1, A_2, \dots, A_n)$ з кортежами даних відповідно. Згідно реляційної теорії кожне відношення повинно відповідати своїй схемі і не мати дублікатів кортежів. За допомогою представ-

лення ресурсів у вигляді відношень можна виконувати типові реляційні операції, длячого за допомогою http запиту типу GET можна вказати необхідні для операції параметри.

За допомогою такого підходу веб-сервіс може надати єдиний API, без необхідності адаптувати його під конкретні потреби клієнтів, а делегує цю задачу на сторону клієнта, який повинен побудувати необхідний йому запит. Таким чином, можна використовувати реляційну теорію у інтерфейсах веб-сервісів на основі мікросервісної архітектури.

Для забезпечення гнучкості інтерфейсу потрібен зручний механізм оновлення даних і на цьому етапі можна виділити методи модифікації ресурсів на рівні сервісу.

Задача формується на умовах оновлення частини ресурсу даних при наявності структури дерева з батьківськи-дочірніми зв'язками між даними. Наведені вимогидо механізму потребують в рамках однієї операції опису певної кількості дій, що дозволяє припустити використання у якості вершини дерева структуру черги, стека або таблиці [3].

Наприклад, для гнучкості інтерфейсу оновлення даних врамках однієї операції можна описати дії, які необхідно виконати над кожним елементомдерева ресурсів. На прикладі вебсервісу магазину, ресурс покупця та йогозамовлення можна представити наступним JSON об'єктом:

{"customer_Name":"J","address":{city:"City_1"},
"orders":[{"order_Name":"Order_1"},{"order_Name":"Order_2"}]}

Цей об'єкт можна розглянути як дерево, кожна з вершин якого описуєназву атрибуту одного із ресурсів та його значення. Отже, для позначення атрибуту, якийнеобхідно обновити, достатньо описати шлях до нього у дереві. Для оновлення даних одного ресурсу достатньо вказати тільки шлях та значення. Але для оновлення колекції ресурсів і забезпечення достатньої гнучкості інтерфейсу необхідно також вказати, як само необхідно оновити дані: додати нові, видалити зайві, абооновити поточні. Розглянемо такі операції: add, replace, remove.

Наприклад, для додання нового замовлення покупцю клієнту веб-сервісі необхідно описати add операцію за шляхом /orders. А для видалення замовлення — операцію remove. Таким чином, можна сформувати гнучкий інтерфейс оновлення дереваресурсів шляхом показу за допомогою шляху атрибуту для оновлення, операціїоновлення та за необхідності параметрів операції. Так, наприклад, операції add та replace потребують конкретного значення, а операція remove — ні.

Операцію оновлення дерева ресурсів можна представити наступним JSON об'єктом:

[{"op":"replace","path":"/customer_Name","value":"Barry"},
{"op":"add","path":"/orders/","value":{"order_Name":"Order_2"}}]

Даний об'єкт вказує набір операцій, які необхідно виконати над кожнимелементом дерева ресурсів, як одну операцію оновлення. У наведеному прикладіоновлюються дані ресурсу покупця та додаються дані до ресурсу замовлень. Такийпідхід, окрім гнучкості, надає ще властивість атомарності до операції модифікації даних.

Особливу увагу можна приділити методаммасштабування програмних інтерфейсів.

Міграція технологій розроблення програмного забезпечення з традиційних моделей в нову хмарну модель приводить до розвитку сервісних пропозицій. Пропозиції різних за назвою веб-служб містять велику кількість спільних характеристик, які є атрибутами архітектури системи: продуктивність, відмовостійкість, масштабованість і надійність.

Платформи віртуалізації представляють собою комп'ютернуІнфраструктуру як Сервіс (Infrastructure-as-a-Service, IaaS). Платформи віртуалізаціїпідсилюють технологіюв центрах обробки даних для надання відповідної послуги клієнтам. Стратегія IaaS забезпечує стандартизовану інфраструктуру, яка оптимізована під вимоги клієнта, та має за мету зосередження навколо моделі надання послуг.

Прикладом моделі надання хмарних обчислень є Платформа як послуга (англ. Platformas a service, PaaS). При цьому підході споживач отримує доступ до використання інформаційно-технологічних платформ. Споживач має можливість використовувати платформи та створювати їх віртуальні екземпляри. При динамічному змінюванні кількості обчислювальних ресурсів, які споживаються, на віртуальних екземплярах розгортається прикладне програмне забезпечення з можливостями розроблення, тестування та експлуатації.

При використанні таких моделей хмарних обчислень мікросервіси системи розміщені в межах одного хмарного провайдеру. Це дає можливість агрегувати API кожного мікросервісу у єдиний високорівневий API на рівні цілої системи.

Серед основних методів побудови такого гнучкого інтерфейсу можна виділити: API Proxy та ApiGateway.

Метод API Proxy полягає у створенні окремого сервісу, який надає високорівневий API, до якого звертається клієнт. Для виконання запиту клієнта Apiproxy сервіс перенапрявляє запити до конкретного мікросервісу або їх групи. При такому підході клієнт не має доступу до API кінцевих мікросервісів. Маршрутизація запитів від клієнта налаштовується у Apiproxy сервісів за допомогою різноманітних правил. Так, можна перенаправляти запит за конкретним URL або усі запити що співвідносяться з певним шаблоном.

Метод ApiGateway теж полягає у створенні високорівневого API за допомогою окремого сервісу. Він може працювати як API proxy, але окрім цього додає ще можливості трансформації даних. До таких трансформацій можна віднести:

- модифікація структури запиту до кінцевого мікросервісу (модифікація URL параметрів, схеми та методу HTTP запиту)
- модифікація відповіді від кінцевого мікросервісу (HTTP статус кодів, імен полів Json/xml об'єкту, додання додаткових метаданих)
- паралельний виклик декількох мікросервісів та агрегація їх відповідей у єдиний результуючий об'єкт
- оркестровці запитів до кінцевих мікросервісів для забезпечення транзакцій запитів клієнта.

Для побудови гнучкого інтерфейсу веб служб можна комбінувати методи для проєктування API систем різного масштабу. Для невеликих систем у складі від 3 до 10 мікросервісів може буде достатньо використати наведені методи з рівня сервісів на чолі з API Proxy. Для більших систем можна створювати так звані модульні API — групувати сервіси у модулі, доступ до яких надається через API proxy. а високорівневий API будувати за допомогою API Gateway, який маніпулює ресурсами через модульні API.

Для реалізації мікросервісів із програмними інтерфейсами побудованих за неведеними методами рекомендовано використовувати архітектурні шаблони. У першу чергу на архітектурні шаблони пов'язані з такими атрибутами якості як зручність обслуговування та сумісність [4]. Серед них: розподіл відповідальності за команди і запити, маршрутизатор на основі локального поділу даних, інтеграція на основі REST, асинхронний обмін повідомленнями та інші.

Отже, використовуючи наведено методи можна побудувати гнучкий інтерфейс веб служб різної складності та розміру.

Висновки

Запропоновано методи побудови та надання гнучких програмних інтерфейсів у мікросервісних системах. Проведений аналіз можливості формування стратегії обрання принципів архітектури мережі при проєктуванні системи для подальшого розроблення технології усунення всіх можливих факторів, які впливають на визначення місця розташування з максимальною точністю.

Досліджені підходи, стандарти, протоколи і формати для обміну даними між мікросервісами.Проведений аналіз передбачає можливості формування стратегії обрання методів побудови та надання гнучких програмних інтерфейсів у мікросервісних системах при проєктуванні системи для подальшого розроблення технології модифікації ресурсів та масштабування системи.

Розглянуті принципипобудовипрограмних інтерфейсів для гнучкого надання і модифікації ресурсів, як на рівні окремих мікросервісів, так і на рівні усієї системи вцілому.

Всі розглянуті вище методи мають свої переваги і недоліки та мають місце в сучасних системах. Вони можуть використовуватися у монолітних, сервіс-орієнтованих і мікросервісних системах різної величини.

Список використаної літератури

1. Білецький А.С. Технологія розробки програмного забезпечення інтерфейсу веб-служб, на основі мікросервісної архітектури // Тези XII Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості і освіті» ДНУЗТ, 2018. С. 85.
2. Білецький А.С. Про надання колекції ресурсів операцій вибірки реляційної алгебри // Тези доповідей конференції за підсумками науково-дослідної роботи Дніпровського національного університету імені Олеся Гончара «Проблеми прикладної математики та комп'ютерних наук», 20219. С. 9.
3. Білецький А.С. Про атомарну модифікацію дерева ресурсів у мікросервісній архітектурі // Тези доповідей конференції за підсумками науково-дослідної роботи Дніпровського національного університету імені Олеся Гончара «Проблеми прикладної математики та комп'ютерних наук». 2021, С. 37.
4. Вальдивия Х.А., Лора-Гонсалес А., Лимон К., Кортес-Вердин К., Очаран-Эрнандес Х.О. Патерни мікросервісной архітектури: багатопрофільний огляд літератури. Труди ИСП РАН, том 33, вып. 1, 2021 г., стр. 81–96. DOI: 10.15514/ISPRAS–2021–33(1)–6

ON METHODS OF FLEXIBLE PROVISION AND MODIFICATION OF RESOURCES IN WEB SERVICE INTERFACES IN MICROSERVICE ARCHITECTURE

Biletskyi A., Bozhukha L.

Abstract

An important issue for the development of information and communication systems is the choice of certain principles of organization, design and development of software systems. In the field of modern web-programming the concept of microservice architecture remains relevant.

The problem with this approach is the lack of standards for the format of messaging between microservices.

Microservice architecture is an architectural style with a set of recommendations and rules. There is currently no single standard for implementing microservice software interfaces, which forces developers to have their own API formats and complicates integration with other web services.

The standardized protocol (SOAP) for data exchange via XML format does not properly represent the concept of microservice architecture due to the large amount of data in the message, which slows down the speed of communication between microservices.

To date, there are three main approaches to designing microservice interfaces: REST, GraphQL and gRPC.

Software developers are almost always faced with the task of identifying methods for building software interfaces in microservice systems and providing flexibility to such designs for further possible modification.

The paper presents a comparative review of existing methods: methods of resource representation and resource modification at the service level, methods of building and scaling software interfaces.

The analysis of the possibility of forming a strategy for choosing the principles of network architecture in the design of the system for further development of technology to eliminate all possible factors that affect the location with maximum accuracy.

Approaches, standards, protocols and formats for data exchange between microservices are investigated. The analysis provides opportunities to form a strategy for choosing methods of building and providing flexible software interfaces in microversion systems in system design for further development of technology for resource modification and scaling of the system.

The principles of building software interfaces for flexible provision and modification of resources, both at the level of individual microservices and at the level of the system as a whole are considered.

All the above methods have their advantages and disadvantages and take place in modern systems. They can be used in monolithic, service-oriented and microservice systems of various sizes.

References

[1] Bilecjkyj A.S. (Desember 13, 2018). Tekhnologhija rozrobky proghramnogho zabezpechennja interfejsu veb-sluzh, na osnovi mikroservisnoji arkhitektury [Web service interface software development technology, based on microservice architecture]. In *Suchasni informacijni ta komunikacijni tekhnologhiji na transporti, v promyslovosti i osviti* (pp. 85). Dnipro: Dnipropetrovsjkyj nacionaljnyj universytet zaliznychnogho transportu imeni akademika V.Lazarjana.

[2] Bilecjkyj A.S. (April 01, 2020). Pro nadannja kolekciji resursiv operacij vybirky reljacijnoji alghebry [On providing a collection of resources for relational algebra sampling operations]. In *Problemy prykladnoji matematyky ta komp'juternykh nauk* (pp. 9). Dnipro: Dniprovsjkyj nacionaljnyj universytet imeni Olesja Ghonchara

[3] Bilecjkyj A.S. (2021). Pro atomarnu modyfikaciju dereva resursiv u mikroservisnij arkhitekturi [On atomic modification of the resource tree in microservice architecture]. In *Problemy prykladnoji matematyky ta komp'juternykh nauk* (pp. 37). Dnipro: Dniprovsjkyj nacionaljnyj universytet imeni Olesja Ghonchara

[4] Valjdyvyja Kh.A., Lora-Ghonsales A., Lymon K., Kortes-Verdyn K., Ocharan-Эrnandes Kh.O. (2021). Paterny mikroservisnoj arkhitektury: baghatoprofiljnyj oghljad literatury [Patterns of microservice architecture: a multidisciplinary review of the literature]. *Trudy YSP RAN,* (1(33)), 81-96. DOI 10.15514/ISPRAS–2021–33(1)–6