УДК 004.415.2
В.В. Никитюк, канд. тех. наук, доц., М.В. Тененський, А.В. Орловська
Тернопільський національний технічний університет імені Івана Пулюя

АНАЛІЗ ВИКОРИСТАННЯ EDA ДЛЯ ВИРІШЕННЯ ПРОБЛЕМ СУЧАСНИХ ЗАСТОСУНКІВ ТА СИСТЕМ

V.V. Nykytyuk, Ph.D., Assoc. Prof., M.V. Tenenskyi, A.V. Orlovska

ANALYSIS OF EDA USAGE FOR SOLVING PROBLEMS OF MODERN APPLICATIONS AND SYSTEMS

Ключові слова: Архітектура, мікросервіси, патерни, подія, програмне забезпечення, EDA
Key words: Architecture, microservices, pattern, event, software, EDA

В наш час будь-яке ПЗ чи застосунок для широкого використання є достатньо складними, а їх розробка часто базується на застосуванні мікросервісного підходу, при якому один застосунок ділять на декілька невеликих та незалежних сервісів (мікросервісів), що спілкуються між собою, утворюючи мікросервісну систему. Відповідно до цього застосунки або їх складові повинні постійно обмінюватись інформацією між собою завдяки REST API або RPC. Такі застосунки повинні впоратися з наступними вимогам: доступність, масштабованість та швидка комунікація сервісів [2]. Доступність полягає в тому, що один екземпляр сервісу функціонував в будь-який момент часу. Тому, масштабованість значною мірою відповідає за продуктивність сервісів, проте враховуючи, що в класичних stateful застосунках її реалізація не є легкою задачею. Щодо комунікації, то в типовому сценарії “запит-відповідь” клієнт блокує виконання будь-яких дій до моменту отримання відповіді на свій запит. В мікросервісній архітектурі може призвести до так званого “Microservices Hell” – ланцюг викликів проходить послідовно через п’ять сервісів з середньою швидкістю обробки в 500 мс. За такої умови блокування клієнту триватиме 2.5 сек., що може бути критичним аспектом, особливо якщо ланцюг викликів, як і час обробки запиту кожним сервісом, буде значно довшим, збільшуючи при цьому час блокування. Саме цю проблему покликана вирішити подієво-орієнтована архітектура (англ. Event-driven architecture, скорочено EDA) [2]. Тому архітектурний шаблон проектування ПЗ, в основі якого лежить створення подій та реагування на них в режимі реального реального часу. Застосунки та їх мікросервіси, спроектовані з дотриманням патернів EDA, в основі своїй повинні складатися з трьох основних частин (див. рис. 1) [3].

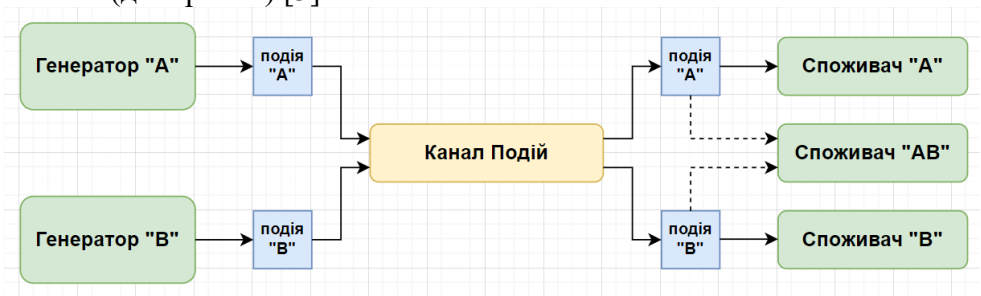

Рисунок 1 – Схематичне зображення функціонування подієво-орієнтованих застосунків

Подією може бути: від запиту користувача на зміну паролю, до аварійного вимкнення одного з сервісів застосунку. Відповідно до рис. 1, генератор подій відповідає за виявлення того чи іншого фактору, на основі якого створює відповідну подію та надсилає її в канал подій. Окрім того, генератор здатний проводити різноманітні операції над даними, трансформуючи їх у зручний для системи вигляд. Канал подій являє собою специфічний механізм, основною задачею якого є забезпечення асинхронної та

безперебійної передачі подій до відповідного заявника. Опрацювання подій відповідає за ідентифікацію події та реагування на неї [4]. Також він може комунікувати з іншими системами чи базою даних для обробки вхідних даних. Оскільки генератор події не очікує відповіді від її споживача, то усувається вищезгадана проблема – блокування виконання коду [1]. Також з рисунку 1 видно, що на одну подію може бути реалізовано декілька споживачів. Такий підхід EDA дозволяє забезпечувати доступність та надійність застосунку. Якщо той чи інший споживач тимчасово недоступний (або не встигає в повній мірі обробити вхідний потік даних) то канал подій, часто представлений сервісами-брокерами, надсилатиме події до іншого споживача, забезпечуючи надійне та стійке до збоїв функціонування системи [2].

Таким чином EDA допомагає реалізувати асинхронну комунікацію сервісів без їх блокування до виконання інших задач. Перехід до EDA означає перехід від класичної stateful моделі, орієнтованої на пряму комунікацію між сервісами, до моделі яка зводиться до комунікації через канал подій, які стають найважливішою складовою обміну інформацією між застосунками [3]. Комбінація підходів EDA з мікросервісною архітектурою дозволяє покращувати масштабованість застосунків шляхом відокремлення комунікації між сервісами від основної бізнес-логіки. Це дозволяє розробляти витривалі до непередбачуваних навантажень системи. Ідеї ПОА можуть бути доповненими принципами SOA (сервіс-орієнтованої архітектури). Дане рішення особливо корисне у випадках, коли споживачі подій розроблюваного застосунку не мають змоги з тієї чи іншої причини забезпечувати виконання власних реакцій [1].

В процесі аналізу використання EDA дозволяє спроектувати та розробити відмовостійкі та оптимізовані застосунки на основі мікросервісної архітектури. Реалізація вільного, неблокуючого зв’язку між сервісами є ключовою перевагою EDA та причиною популяризації її використання. Для підвищення ефективності та надійності застосунку, рекомендується створення більше одної операції на кожний з типів подій. Особливу увагу варто звертати і на безпеку даних, які передаються. Тому перехід до EDA, за умови дотримання усіх вищеописаних принципів, підніме ваш застосунок на якісно новий рівень. Проте не варто очікувати, що EDA вирішить абсолютно усі проблеми комунікації між застосунками та його складовими, адже в деяких випадках все ще можуть знадобитись звичайні виклики по REST API типу “запит-відповідь”.

Література

1. Cameron D. Complex Event Processing and Service oriented Architecture (SOA). TMCNET NEWS. August 18, 2006. URL: https://www.tmcnet.com/usubmit/2006/08/18/1816129.htm

2. Sucaciu B. How event-driven architecture solves modern web app problems. Stackoverflow Blog. 2020, March 16. URL: https://stackoverflow.blog/2020/03/16/how-event-driven-architecture-solves-modern-web-app-problems/

3. Welsh M., Brewer E. SEDA: an architecture for well-conditioned, scalable internet services. ACM SIGOPS Operating Systems Review. 2001. Vol. 35, Issue 5. P. 230–243. URL: https://dl.acm.org/doi/10.1145/502059.502057

4. V. Nykytyuk, V. Dozorskyi, O. Dozorska, A. Karnaukhov and L. Matiichuk. The Method of User Identification by Speech Signal. The 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP-2022) Ternopil, Ukraine, November 22-24, 2022. Vol-3309 urn:nbn:de:0074-3309-1. P.225-232. ISSN 1613-0073 DOI: 10.1425/jsdtl. (Scopus).