УДК 004.75

Н. С. Жмуцька

ПОБУДОВА ВЕБ-ДОДАТКІВ НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Вінницький національний технічний університет

Анотація

Розглянуто метод побудови веб-додатків на основі мікросервісної архітектури. Визначено переваги та недоліки використання даного архітектурного стилю.

Ключові слова: веб-додаток, мікросервіс, архітектура, сервіси, протокол.

Abstract

The method of constructing web-based applications based on microservice architecture is considered. The advantages and disadvantages of using this architectural style are determined.

Keywords: web application, microservice, architecture, services, protocol.

Вступ

Термін «Мікросервісна архітектура» набув поширення в останні кілька років як опис способу дизайну додатків у вигляді набору незалежно розгорнутих сервісів. На жаль, існує не так багато інформації, яка описує, що таке мікросервіси і як застосовувати їх.

Метою роботи є огляд методу побудови веб-додатків на основі мікросервісної архітектури.

Результати дослідження

Архітектурний стиль мікросервісів - це підхід, при якому єдиний додаток будується як набір невеликих сервісів, кожен з яких працює у власному процесі і взаємодіє з іншими використовуючи легковагі механізми, як правило HTTP. Ці сервіси побудовані навколо бізнес-потреб і розгортаються незалежно з використанням повністю автоматизованого середовища. Існує абсолютний мінімум централізованого управління цими сервісами. Самі по собі ці сервіси можуть бути написані на різних мовах і використовувати різні технології зберігання даних.[1]

Архітектура мікросервісів використовує бібліотеки, але їх основний спосіб розбиття додатку - шляхом ділення його на сервіси. Бібліотеки визначаються як компоненти, які підключаються до програми і викликаються нею в тому ж процесі, в той час як сервіси - це компоненти, що виконуються в окремому процесі і взаємодіють між собою через веб-запити або remote procedure call (RPS).

Головна причина використання сервісів замість бібліотек - це незалежне розгортання.

Інший наслідок використання сервісів як компонентів - більш явний інтерфейс між ними. Більшість мов програмування не мають хорошого механізму для оголошення Published Interface. Часто тільки документація і дисципліна запобігають порушення інкапсуляції компонентів. Сервіси дозволяють уникнути цього через використання явного механізму віддалених викликів.

Тим не менше, використання сервісів подібним чином має свої недоліки. Дистанційні виклики працюють повільніше, ніж виклики в рамках процесу, і тому API повинен бути менш деталізованим (coarser-grained), що часто призводить до незручності у використанні.

Мікросервісний підхід до розбиття має на увазі розбиття на сервіси відповідно до потреб бізнесу. Такі сервіси включають в себе повний набір технологій, необхідних для цієї бізнес-потреби, в тому числі для користувацький інтерфейс, сховище даних і будь-які зовнішні взаємодії. Це призводить до формування крос-функціональних команд, що мають повний набір необхідних навичок.[2]

Додатки, створені з використанням мікросервісної архітектури, мають тенденцію бути настільки незалежними (роз'єднаними) і сфокусованими (зв'язковими), наскільки це можливо: вони містять

свою власну доменну логіку і діють більше як фільтри в класичному сенсі Unix - вони приймають запити, застосовують логіку і відправляють відповідь. Замість складних протоколів, таких як WS- * або BPEL, вони використовують прості протоколи на основі REST.

Двома найбільш часто використовуваними протоколами є HTTP-запити через API ресурсів і спрощений сеанс повідомлень.

Команди, що практикують мікросервісну архітектуру, використовують ті ж принципи і протоколи, на яких будується всесвітня павутина (і фактично Unix). Часто використовувані ресурси можуть бути кешованими без особливих зусиль з боку розробників або IT-адміністраторів.

Другим найбільш часто використовуваним засобом зв'язку є полегшена шина повідомлень. Така інфраструктура, як правило, не містить доменної логіки - прості реалізації, такі як RabbitMQ або ZeroMQ, нічого не роблять, крім як забезпечують асинхронну фабрику. Логіка в той же час існує на кінцях шини - в сервісах, які відправляють і отримують повідомлення.

Наслідком використання сервісів як компонентів є необхідність проектування додатків так, щоб вони могли працювати при відмові окремих сервісів. Будь-яке звернення до сервісу може не спрацювати через його недоступність. Клієнт повинен реагувати на це настільки терпимо, наскільки можливо. Це є недоліком мікросервісів в порівнянні з монолітом, тому що це вносить додаткову складність в додаток.

.

Висновки

Встановлено, що мікросервісний стиль - важлива ідея, що варта розгляду. Існує безліч організацій, які вже давно використовують те, що називається мікросервісами, але не використовують цю назву.

Незважаючи на весь позитивний досвід, не можна стверджувати, що мікросервіси - це майбутнє проектування ПО.

Успіх будь-яких спроб побудувати компонентну систему залежить від того, наскільки добре компоненти підходять додатку. Складно зрозуміти де саме повинні лежати кордони компонентів.

Ще одна проблема полягає в тому, що якщо компоненти не підібрані досить правильно, відбувається перенесення складності з компонентів на зв'язки між компонентами. Створюється хибне відчуття простоти окремих компонентів, в той час як вся складність знаходиться в місцях, які важче контролювати.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ньюмен С. Створення мікросервісів / С. Ньюмен. – Санкт-Петербург, 2016. – 304 с.

2. Tilkov, Stefan (17 November 2014). "How small should your microservice be?". innoq.com. Retrieved 4 January 2017.

3. Азарова А.О. Методичні вказівки до проведення практичних занять та до виконання самостійної індивідуальної роботи з дисципліни «Основи науково-дослідної роботи» для студентів напрямів підготовки 6.030601 – «Менеджмент» та 6.170103 – «Управління інформаційною безпекою» / Азарова А.О., Карпінець В.В. – Вінниця: ВНТУ, 2013. – 44 с.

Наталія Станіславівна Жмуцька – студентка групи УБ-18м, факультет менеджменту та інформаційної безпеки, Вінницький національний технічний університет, м. Вінниця, e-mail: nataliiazhm@gmail.com

Науковий керівник: Людмила Миколаївна Ткачук – канд. економічних наук, доцент кафедри МБІС, Вінницький національний технічний університет, м. Вінниця, e-mail: ludatkachuk2017@gmail.com

Nataliia Zhmutska - student of UB-18m group, faculty of Management and Information Security, Vinnitsa National Technical University, Vinnitsa, e-mail: nataliiazhm@gmail.com

Supervisor: Liudmyla Tkachuk – cand. economic sciences, Assistant Professor of Building MBIS, Vinnitsa National Technical University, Vinnitsa, e-mail: ludatkachuk2017@gmail.com