УДК 621.326

Дар'я Макарова, студентка
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

АРХІТЕКТУРА МІКРОСЕРВІСІВ

Сьогоднішні підприємства стикаються з багатьма проблемами які орієнтовані на клієнта в розподільних система. Парадигма мікросервісів стала «наступною великою річчю» для надання IT-результатів для підтримки сучасного підприємства, з безліччю постачальників технологій і послуг. Цей документ надає собою набір рекомендацій та кращих практик щодо ефективного використання та управління програмними компонентами підприємства, грунтуючись на кращих концепціях і практиці мікросервісної парадигми розробки програмного забезпечення.
Ключові слова: мікросервісна архітектура, архітектура розробки ПЗ, моноліт, стандарти.

Darya Makarova

MICROSERVICES ARCHITECTURE

Today's Enterprises are facing many challenges in the service oriented, customer experience centric and customer demand driven global environment. Microservice paradigm has emerged as the "next big thing" for delivering IT outcomes to support the modern enterprise, with many technology vendors and. This paper presents a set of recommendations and best practices on the effective use and management of enterprise software components, drawing upon the best of Microservice concepts and practice.
Key words: microservices architecture, computer architecture, monolothic architecture, standards.

Більшість сучасних програм підтримують безліч різних клієнтів, в тому числі настільних браузерів, мобільних браузерів і нативних мобільних додатків. Їм також необхідно інтегруватися з іншими додатками, або через веб-службу або брокер повідомлення. Додаток обробляє запити для виконання бізнес-логіки; доступ до бази даних; обмінюється повідомленнями з іншими системами і повертає відповіді клієнту у вигляді сформованої сторінки, яка містить в собі дані з множин сервісів.

Рішення

Нововведенням в світі розробки тяже ПО, яке вирішує більшість проблем з якими стикаються сучасні проекти стала архітектура мікросервісів Концепція якої полягає в тому, що б розбити моноліт з єдиним ядром, на сервіси, кожен з яких відповідає за свою бізнес-логіку і при необхідності його легко видозмінити без втрати великої кількості часу і без простою всього проекту.

Переваги:

- Кожен microservice є відносно невеликим:
  - Простіше для розробників, щоб зрозуміти,
  - Інструментальна швидше робить розробників більш продуктивними

- Додаток запускається швидше, що робить розробник більш продуктивним, і прискорює розгортання;

• Слабка міжсервісна залежність. Кожна команда може розробляти, розгортати і масштабувати свої послуги незалежно від всіх інших команд.

• Покращена ізоляція несправностей. Вихід з ладу одного з сервісів не впливає на роботу інших. Інші послуги будуть продовжувати обробляти запити. Для порівняння, збій одного з компонентів монолітної архітектурі може зупинити всю систему.

• Кожна послуга може бути розроблена і впроваджена незалежно один від одного

• Слабка залежність від стандартів і технологій. При розробці нового сервісу можна вибрати новий набір технологій. Також при внесенні суттєвих змін в існуючу послугу можна переписати його, використовуючи інший набір технологій.

Недоліки:

• Інструменти розробника орієнтовані на будівництво монолітних додатків і не забезпечують явну підтримку для розробки розподілених систем;

• Складність тестування окремих сервісів;

• Реалізація сервісів, які охоплюють кілька послуг вимагає ретельної координації між командами;

• Складність розгортання;

• Підвищене споживання пам'яті. Кожна служба працює у своїй власній віртуальній машині, якії необхідні свої ресурси.

Коли слід використовувати архітектуру мікросервісів?

Одна з проблем, з використанням цього підходу полягає у визначенні, коли має сенс використовувати цю архітектуру. Складна розподілена архітектура, уповільнює розвиток. Це є серйозною проблемою для стартапів, їх вузьким місцем є те, що вони швидко розвиваються. Коли завдання полягає в тому, як масштабувати і використовувати функціональну декомпозицію, заплутані залежності ускладнюють декомпозицію монолітного додатку на набір послуг.

Як розбити додаток на окремі сервіси?

Ще одна проблема полягає у визначенні, як розбити систему на мікросервіси. Для цього є цілий ряд стратегій слідуючи яким можна досягти бажаного результату:

• Розкласти по бізнес процесам і визначити послуги, відповідні бізнес - можливостям.

• Розкласти по доменним областям.

• Розкласти по послугах, які відповідають за конкретні дії.

• Розкласти шляхом визначення служби, які відповідає за всі операції з юридичних ресурсів даного типу.

Ще одна аналогія, яка допомагає при проектуванні послуг є розробка Unix утиліт. Unix надає велику кількість утиліт. Кожна утиліта робить рівно однин функціонал і може бути об'єднана з іншими, використовуючи скрипт для виконання складних завдань.

Як зберегти цілісність даних?

Для того, щоб забезпечити слабкий зв'язок між сервісами необхідно, що б кожна послуга мала свою власну базу даних. Підтримка цілісності даних між службами є однією з головних проблем. Мікросервісна архітектура підтримує кілька способів

надійного відновлення даних і публікації подій, включаючи Event Sourcing і журнал транзакції.

Практика відомих компаній.

Більшість великомасштабних веб - сайтів, включаючи Netflix, Amazon і eBay еволюціонували від монолітної архітектури до архітектури мікросервісов.

Netflix, який є дуже популярним відео сервісом, який відповідає за 30% інтернет-трафіку, має великий масштаб, сервіс-орієнтованої архітектури. Вони обслуговують понад мільярд дзвінків в день з більш ніж 800 різних видів пристроїв.

Amazon.com спочатку мав дворівневу архітектуру. Після збільшення вони мігрували в сервіс-орієнтовану архітектуру, що складається з сотень серверних послуг. На сайті Amazon.com додаток контактує з 100-150 послугами, щоб отримати дані, які використовується для створення веб-сторінки.

Аукціон ebay.com сайт також еволюціонували від монолітної архітектури до сервіс-орієнтованої архітектури. Рівень додатків складається з декількох незалежних сервісів. Кожна програма реалізує бізнес-логіку для конкретної функціональної області, як покупка або продаж.

Є також безліч інших прикладів компаній, що використовують архітектуру мікросервісів.

Підводячи підсумки, то я хотіла б підкреслити, що моя основна мета при написанні цієї статті полягала в тому, щоб пояснити основні ідеї і принципи мікросервісной архіктури. Я вважаємо, що мікросервісний стиль - важлива ідея, що стоїть для розгляду для enterprise додатків. Ця архітектура була успішна протестіровна на чималій кількості успішних компаній і швидко прижилася в сфері IT.

Література

1. Никита Цуканов. Архитектура микросервисов: [Электронный ресурс]: URL: http://itnan.ru/post.php?c=1&p=320962

2. Partitioning Problems in Parallel, Pipelined, and Distributed Computing / Bokhari_S. // IEEE Transactions Computers. –1988 – 57 с.

3. Алексей А.О. Переход от монолита к микросервисам [Электронный ресурс]: URL: https://habrahabr.ru/post/305826/.

4. Никита Цуканов. Проектирование и микросервисы для самых маленьких [Электронный ресурс]: URL: https://habrahabr.ru/post/311208/

5. Ньюмен С. Создание микросервисов. – СПб.: Питер, 2016. – 304 с.: ил. – (Серия «Бестселлеры O’Reilly»).