Максимов Олександр Семенович, старший викладач
Одеський національний університет імені І.І.Мечникова, Одеса
Кравченко Кирил Дмитрович,
студент 3 курсу, спеціальності інформаційні системи та технології
Одеський національний університет імені І.І.Мечникова, Одеса

ПОБУДОВА ТА ЗАБЕЗПЕЧЕННЯ ФУНКЦІОНУВАННЯ СЕРВІСІВ МІКРОСЕРВІСНОЙ АРХІТЕКТУРИ.

Актуальність дослідження пов'язана з тим, що в останні кілька років «Microservice Architecture» набув поширення як опис способу дизайну додатків у вигляді набору незалежно розгорнутих сервісів. У той час як немає точного опису цього архітектурного стилю, існує загальний набір характеристик: організація сервісів навколо бізнес-потреб, автоматичне розгортання, перенос логіки від шини повідомлень до приймачів (endpoints) і децентралізований контроль над мовами і даними [2].

Слід зазначити, що архітектурний стиль мікросервісів - це підхід, при якому єдине додаток будується як набір невеликих сервісів, кожен з яких працює у власному процесі і коммунікує з іншими використовуючи легковагі механізми, як правило HTTP. Ці сервіси побудовані навколо бізнес-потреб і розгортаються незалежно з використанням повністю автоматизованої 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. [3]

Для того, щоб почати розповідь про стилі мікросервісів, найкраще порівняти його з монолітом (monolithicstyle): додатком,

побудованому як єдине ціле. Enterprise додатки часто включають три основні частини: призначений для користувача інтерфейс (що складається в основному з HTML сторінок і javascript- а), база даних (як правило реляційної, з безліччю таблиць) і сервер. Серверна частина обробляє HTTPзапити, виконує доменну логіку, запитує і оновлює дані в БД, заповнює HTMLсторінки, які потім відправляються браузеру клієнта. Будь-яка зміна в системі призводить до перезібравши і розгортання нової версії серверної частини програми [1].

Монолітний сервер - досить очевидний спосіб побудови подібних систем. Вся логіка по обробці запитів виконується в єдиному процесі, при цьому ви можете скористатися наявними можливостями вашого мови програмування для поділу додатки на класи, функції і namespace-и. Ви можете запускати і тестувати додаток на машині розробника і використовувати стандартний процес розгортання для перевірки змін перед публікацією їх в продакшн [2].

Монолітні додатки можуть бути успішними, але все більше людей розчаровуються в них, особливо в світлі того, що все більше додатків розгортаються в хмарі. Будь-які зміни, навіть самі невеликі, вимагають пересборки і розгортання всього моноліту. З плином часу, стає важче зберігати хорошу модульну структуру, зміни логіки одного модуля мають тенденцію впливати на код інших модулів. Масштабувати доводиться все додаток цілком, навіть якщо це потрібно тільки для одного модуля цього додатка [3].

Ці незручності призвели до архітектурного стилю мікросервісів: побудови додатків у вигляді набору сервісів. На додаток до можливості незалежного розгортання і масштабування кожен сервіс також отримує чітку фізичну межу, яка дозволяє різним сервісам бути написаними на різних мовах програмування. Вони також можуть розроблятися різними командами.

Архітектура мікросервісів використовує бібліотеки, але їх основний спосіб розбиття додатку - шляхом ділення його на сервіси. Головна причина використання сервісів замість бібліотек - це незалежне розгортання.

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. Якщо вам потрібно змінити набір відповідальностей між компонентами, зробити це складніше через те, що вам потрібно перетинати кордони процесів.

Список використаних джерел

1. Microservices URL: https://martinfowler.com/articles/microservices.html (Last accessed: 11.03.2021).
2. Микросервисы (Microservices) URL: https://habr.com/ru/post/249183/ (дата звернення: 11.03.2021).
3. Общие сведения о Service Fabric URL: https://docs.microsoft.com/ru-ru/azure/service-fabric/service-fabric-overview (дата звернення: 11.03.2021).