Макарова Д.О.
студентка,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ПОРІВНЯЛЬНИЙ АНАЛІЗ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ І МОНОЛІТНОЇ АРХІТЕКТУРИ

На даний час накопичений значний досвід в розробці розподілених систем, який дозволяє говорити про наявність типових архітектур.

Класифікація архітектур і критерії, що їх характеризують представлені в таблиці 1.

Таблиця 1

Порівняння архітектур проектування інформаційних систем

Критерій / Архітектура	Монолітна архітектура	Архітектура клієнт-сервер	Веб-орієнтована архітектура	Сервіс-орієнтована архітектура	Мікросервісна Архітектура
Простота реалізації	+	+	+	-	-
Узгодження даних	+	+	-	-	-
Модульність	-	+	+	+	+
Простота рефакторінга	-	-	-	+	+
Кроссплатформеність	-	-	-	+	+
Масштабування	-	+	+	+	+
Відмовостійкість	-	-	-	+	+

Джерело: розробка автора

При уважному вивченні гетерогенних систем, можна помітити досить багато спільного в їх будові. Якщо розглядати такий критерій для порівняння як модульність, то градація моделей проходить між двома крайнощами, монолітна і мікросервісна архітектури. Є цілком монолітні системи, коли весь проект сконцентрований в одному модулі. Сумішшю монолітної і мікросервісної архітектури є комбіновані системи, коли деякий функціонал виноситися з моноліту, як окремий мікросервіс не втрачаючи з ним зв'язок, і є мікросервісна модель, в такому випадку моноліт повністю розділений на окремі сервіси.

Для того, щоб почати порівняльний аналіз даних парадигм побудови ПЗ, необхідно розглянути дві ці архітектури окремо.

Архітектурний стиль мікросервісів – модель при якій єдиний модуль розбивається на набір невеликих сервісів, кожен з яких незалежний і працює в окремому процесі. При цьому кожен сервіс відображає окрему бізнес-потребу і

розгортаються незалежно з використанням повністю автоматизованого середовища.

Монолітом – називається програмний продукт, який побудовано як єдине ціле. Кожен рефакторінг коду призводить до повної перезбірки і розгортання нової версії серверної частини програми. Вся логіка по обробці запитів виконується в єдиному процесі, при цьому можна використовувати можливості мови програмування для поділу додатка на класи, функції та бібліотеки.

Порівнювати між собою такі архітектури, як монолітний Enterpise проект і мікросервісну архітектуру, і достовірно визначити яка з них більш виграшна, яка з них принесе компанії менше проблем і більше прибутку, без контексту компанії, і її бізнес завдань неможливо. Обидві мають ряд переваг і недоліків, і кожен з них буде мати різну вагу для різних систем. Іноді недолік може стати перевагою і навпаки.

Таблиця 2

Порівняння мікросервісної архітектури і моноліту за критеріями Мартіна Фаулера

Критерій/Архітектура	Моноліт	Мікросервіси
Простота	Простіше в реалізації, в управлінні і в розгортанні, і вимагає меншої підготовки у команди розробників.	Потребують більш ретельного управління, оскільки вони розгортаються на різних серверах і використовують API.
Узгодженість	Спрощує підтримку узгодженості коду, тому що все зберігається в загальній базі.	Підтримувати узгодження даних складніше, тому що система управляється різними командами з дотриманням різних стандартів.
Межмодульний рефакторінг	Забезпечує просту і швидку взаємозв'язок між модулями і переміщення класів з одного модуля в інший.	Необхідно дуже чітко визначати межі модулів.
Доступність	При виході з ладу хоча б одного компонента, може постраждати вся система.	Компоненти не так жорстко пов'язані між собою і являють собою самостійні незалежні елементи.
Кросплатформеність	Повна залежність від платформи і мови програмування.	Є можливість використовувати різні технології і середовища відповідно до заявлених бізнес-процесами.

Джерело: розробка автора

На чию користь віддати свій вибір, залежить від ряду критеріїв, від оцінки факторів, які є критичними для певної системи і того, як вони впливають на результати. Мартін Фаулер, автор ряду книг і статей про архітектуру ПЗ, виділяє критерії відштовхуючись від яких можна обрати більш кращу модель для певної системи. Ці критерії наведені у таблиці 2.

На основі викладеного можна зробити такі висновки, що мікросервіси більш трудомісткі, і це окупається тільки для більш складних систем. Тому, якщо можна впоратися зі складністю системи за допомогою монолітної архітектури, не слід піддаватися впливу моди і переходити на мікросервіси. Тому багато провідних фахівців радять починати з монолітної системи і переходити на мікросервіси тільки в тому випадку, якщо ви абсолютно впевнені, що без поділу на сервіси проблему ніяк не вирішити. Наприклад, якщо вартість додавання нового функціоналу починає перевершувати вигоду від цього самого функціоналлу, то слід задуматися про складність рефакторінга великого моноліту і почати планувати перехід до мікросервісам.

Список використаних джерел:

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»).

Обловацька М.В.

студент,

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

ПОРІВНЯННЯ НАТИВНОЇ ТА КРОСПЛАТФОРМНОЇ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

Нативна (від англ. native – рідний, природний) розробка передбачає використання оригінальних мов та інструментів розробки тієї чи іншої операційної системи. Наприклад, додатки для Android створюються в середовищі Android Studio за допомогою мови Java, а для iOS – у середовищі XCode за допомогою мов Objective-C, Swift, C та C++ відповідно. Тобто нативними додатками можна назвати ті, з якими користувач зустрічається з першого дня використання пристрою. Такі застосування можуть отримувати доступ до усіх служб та сервісів телефону.