МИКРОСЕРВИСНАЯ АРХИТЕКТУРА

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Криволап Д.Э.

Терех И. С. – кандидат технических наук

В современных приложениях при росте количества пользователей и количества запросов на сервер возрастают и требования к отказоустойчивости , высоким нагрузкам и маштабируемости. При этом монолитная архитектура не расчитана на высокие нагрузки, поэтому приходит на помощь микросервисная архитектура.

Применение нового вида архитектуры несет за собой плюсы и минусы. Переход с монолитной архитектуры на микросервисную стоит хорошо обдумать. Для сравнения проанализируем сервис-ориентированную и микросервисную архитектуры.

Параметр	**SOA**	**Микросервисы**
Межсервисное взаимодействие	Умные каналы, такие как сервисная шина предприятия, с использованием тяжеловесных протоколов вроде SOAP и других веб-сервисных стандартов	Примитивные каналы, такие как брокер сообщений, или прямое взаимодействие между сервисами с помощью легковесных протоколов наподобие REST или gRPC
Данные	Глобальная модель данных и общие БД	Отдельные модель данных и БД для каждого сервиса
Типовой сервис	Крупное монолитное приложение	Небольшой сервис

Таблица 1

Некоторые критики микросервисов утверждают, что в этом подходе нет ничего нового и что это всего лишь разновидность сервис-ориентированной архитектуры (service-oriented architecture, SOA). Действительно, на самом высоком уровне существует некоторое сходство. И SOA, и микросервисная архитектура – это стили проектирования, которые структурируют систему как набор сервисов. Но при более детальном рассмотрении можно обнаружить существенные различия (таблица 1).

SOA и микросервисная архитектура обычно используют разные стеки технологий. В приложениях на основе SOA, как правило, применяются тяжеловесные стандарты веб-сервисов наподобие SOAP. Им часто нужна сервисная шина предприятия (Enterprise Service Bus, ESB) — умный канал, который интегрирует сервисы с помощью бизнес-логики и кода для обработки сообщений. Приложения, спроектированные в виде микросервисов, обычно задействуют легковесные технологии с открытым исходным кодом. Сервисы взаимодействуют через примитивные каналы, такие как брокеры сообщений или простые протоколы, подобные REST или gRPC. SOA и микросервисная архитектура также по-разному обращаются с данными. SOA-приложения обычно имеют глобальную модель данных и общую БД а у каждого микросервиса есть собственная база данных.

Проектирование связано с определением компромиссов и приоритетов. Микросервисная архитекрура помимо своих достоинств имеет недостатки такие как сложности при реализации распределенных транзакций и более затруднительным взаимодействием при тестировании.

В идеальном мире все системы обладали бы бесконечным быстродействием, не предъявляли никаких требований к подсистеме хранения данных, давали нулевую нагрузку на сеть, никогда не содержали никаких ошибок и создавались без всяких затрат. Однако в реальном мире один из важнейших аспектов работы проектировщика — анализ конкурирующих характеристик проекта и достижение баланса между ними. Если быстрота отклика системы важнее, чем минимизация времени разработки, проектировщик выберет один вариант. Если во главе угла быстрота разработки, оптимальным может оказаться другой вариант проекта.

Список использованных источников:

1. Крисс Ричардсон. Книга «Микросервисы. Паттерны разработки и рефакторинга» // Крисс Ричардсон. – Питер, 2019. – 544 с
2. Стив Макконнелл. Совершенный код. Мастер-класс // Издательство «Русская редакция», 2010. — 896 стр