Столыпинский
вестник

Научная статья
Original article
УДК 004.031

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МИКРО-СЕРВИСНОЙ И МОНОЛИТНОЙ АРХИТЕКТУРЫ

COMPARATIVE ANALYSIS OF MICRO-SERVICE AND MONOLITHIC ARCHITECTURE

Бежик А.А., Студент бакалавриата 3 курс, МИРЭА - Российский технологический университет (РТУ МИРЭА)
Мажей Я.В., Студент бакалавриата 3 курс, МИРЭА - Российский технологический университет (РТУ МИРЭА)

Bezhik A.A., 3rd year undergraduate student, MIREA - Russian Technological University (RTU MIREA)
Mazhey Ya.V., 3rd year undergraduate student, MIREA - Russian Technological University (RTU MIREA)

Аннотация: В наше время остро стоит вопрос об выборе архитектуры приложений. Основные архетипы современной архитектуры показывают абсолютно противоположные взгляды на данный вопрос. Для оценки лучшего взгляда необходимо провести анализ данных типов архитектур.

Abstract: Nowadays, the question of choosing an application architecture is acute. The main archetypes of modern architecture show absolutely opposite views on this issue. To assess the best view, it is necessary to analyze these types of architectures.

Ключевые слова: Архитектура, архетипы, микро-сервисная, монолитная.

Keywords: Architecture, archetypes, micro-service, monolithic.

Настоящая статья посвящена теме что такое архитектура, зачем она нужна, её виды, критерии хорошей архитектуры.

В какой-то момент времени большинство мировых гигантов IT индустрии были зависимы в разработке от монолитной архитектуры. Данный метод широко использовался так как был одним из самых удобных для того, чтобы соответствовать рынку и выполнять возложенные на них задачи. Однако метод монолитной разработки создавал множественные риски при расширении функционала основного ПО.

Монолитные архитектуры нельзя охарактеризовать как плохие, ведь на практике они часто становятся очень удобным и одним из лучших вариантов для организаций на ранних этапах их развития. Согласно высказыванию инженера и программиста Рэнди Шупа: «нет идеальной архитектуры для всех продуктов и всех масштабов. Любая архитектура соответствует некоторому набору целей или диапазонам требований и ограничений, таких как время вывода на рынок, простота развития функциональности, масштабирование и так далее. Функциональность любого продукта или услуги почти наверняка будет изменяться с течением времени, так что нет ничего удивительного, что наши архитектурные потребности будут меняться. Что работает в масштабе 1x, редко работает в масштабе 10x или 100x». [Книга DevOps]

Рассмотрим архитектурные архетипы по подробнее (Таблица 1).

Таблица 1 – Архитектурные архетипы

Тип архитектуры	**Достоинства**	**Недостатки**
Монолитная архитектура версии 1 (весь функционал в единой монолитной структуре)	Довольно простая в начальном написании, Низкие задержки между взаимодействиями разных процессов, Один блок развертывания (проект) и одна кодовая база, В маленьких масштабах позволяет эффективно использовать ресурсы.	Довольно трудно и денежно затратно при попытке расширения функционала и по мере увеличения количества человек, работающих над ПО, Отсутствует поддержка модульности, так как весь код - единый монолит, Плохая масштабируемость,

		Поскольку весь код пишется с нуля, более затратно по времени.
Монолитная архитектура версии 2 (весь функционал разделен на набор монолитных слоев, каждый из которых отвечает за свою часть)	Довольно простая в начальном написании, Легче присоединять к запросам, Единый блок развертывания (проект) и единая кодовая база, В маленьких масштабах позволяет эффективно использовать ресурсы.	Увеличение количества внутренних связей между слоями с течением времени разработки, Избыточность деления по слоям и как следствие плохая масштабируемость, Трудная настройка для персонала незнакомым с деталями разработки и как следствие большая трудность в поддержке сторонними разработчиками, Построена на схеме управления «все или ничего».
Микросервисная архитектура	Каждый микросервис, отвечающий за свои функции прост в освоении и интеграции в единую систему микросервисов, Масштаб и производительность не зависят от требуемого функционала и тех задания, Отсутствие зависимости в ходе тестирования и развертывания ПО, Возможна оптимальная настройка производительности по принципу «разделяй и властвуй»	Необходимость наладить множественное сотрудничество между подразделениями, Наличие множества репозиториев с микросервисами, Более емкостные требования к более сложным инструментам и отладке зависимостей между микросервисами, Задержки сетевых сервисов, которые могут возникать в ходе работы/разработки

Из таблицы видно, что у каждого из данных архетипов есть как свои достоинства, так и свои недостатки. Выбор архитектуры организацией зависит только от её эволюционных потребностей, выбираемых на основе достоинств и недостатков типов архитектур. Судя по данным из таблицы монолитная архитектура более удобная для маленьких проектов и для организаций типа «стартап», и это в корень отличается от микро-сервисной архитектуры, поскольку она требует наличия множества команд разработчиков, каждая из

которых отвечает за внедрение своего функционала и несет единоличную ответственность за представление своей части продукта клиенту.

Пример эволюционной эксплуатации архетипов архитектур: переход от монолитной к микросервисной

Как было выяснено ранее эволюционный переход компаний от монолитной к микросервисной ни что иное как закономерное развитие внутри компании с течением времени, поскольку её инфраструктуре с течением времени необходимо расширятся и как следствие уходить от монолитной модели разработки к микросервисной.

Одним из наиболее ярких и изученных преобразований такого типа можно назвать эволюционный путь ИТ гиганта Amazon. В интервью с одним из ведущих технических специалистов компании Microsoft Джимом Греем Вернер Фогельсом, технический директор компании Amazon рассказал что сервис amazon.com начал свою работу в 1996 г. в виде «монолитного приложения на веб-сервере, обращаясь к базе данных на серверной части. Это приложение, получившее название Obidos, эволюционировало, чтобы сохранить всю бизнес-логику, всю логику отображения и все функциональные возможности, которые в конце концов прославили Amazon: аналоги, рекомендации, Listmania, обзоры и так далее»

Со ходом развития компании данное приложение разрослось до огромных масштабов и стало слишком запутанным, в котором невозможно было выделить отдельные части в ходе их взаимодействия (настолько сильно они были переплетены). В следствии чего приложение попросту было невозможно масштабировать по мере необходимости. По мнению Фогельса это значило что многие вещи которые могли бы быть сделаны в хорошо масштабируемой программной среде попросту более не могли быть реализованы, и как следствии развитие данного программного комплекса попросту было невозможно.

Придя к этому выводу руководству амазон пришлось начать развивать новую архитектурную модель, которая будет отвечать заданным функциональным требованиям и при этом соответствовать необходимому

масштабу. По словам Грея: «Мы прошли через период серьезного самоанализа и пришли к выводу о том, что сервис-ориентированная архитектура могла бы обеспечить уровень изоляции, достаточный для того, чтобы позволить нам создавать множество компонентов программного обеспечения быстро и независимо друг от друга» [Книга по DevOps]

Таким образом компания Amazon была вынуждена пройти большой путь по изменению архитектуры, для того чтобы сменить двухуровневую монолитную архитектуру, которую они использовали с момента основания на полностью разделенную с точки зрения функционала платформу сервисов, которые были децентрализованы и могли обслуживать множество различных приложений в единицу времени вне зависимости от масштаба требуемой задачи : «Потребовалось множество инноваций, чтобы такие изменения стали возможными, и мы были одними из первых, использовавших этот подход», говорил Фогельс [Книга по DevOps]

На основе работы, проделанной внутри компании, были сделаны следующие выводы:

– При необходимости достижения изоляции стоит использовать сервисы, ведь это даст самый высокий уровня владения и управления

– Для высокой масштабируемости и повышения надежности сервиса необходимо запретить прямой доступ клиентов к базе данных

– Сервисно ориентированная модель в масштабах огромных ИТ корпораций обладает рядом преимуществ, например: создание и развитие отдельных команд разработчиков для быстрого создания инноваций в интересах клиентов

Данный путь развития команды Amazon позволил повысить продуктивность и надежность работы с клиентами до показателей несравнимых с показателями приложения построенного на монолитной архитектуре (в 2015 г. сервисы Amazon выполняли почти 136 тыс. развертываний в день, в то время как в 2011 всего 15 тысяч).

Вывод о монолитной структуре приложения в информационном вакууме

После рассмотрения монолитной структуры приложения на контрасте с микросервисной и приведения примера разработки монолитной структуры и её эволюции в микросервисную необходимо рассмотреть монолитную архитектуру в отдельности от всех остальных альтернатив. Это необходимо для того, чтобы сделать объективную оценку данной архитектуры по следующим параметрам:

- Эффективность
- Гибкость
- Расширяемость
- Масштабируемость
- Тестируемость
- Повторное использование

Для начала рассмотрим эффективность данной архитектуры приложения. Монолитная архитектура чаще всего выбирается при разработке не больших стар-тап приложений, когда у организации нет возможности выделить множество команд для работы с отдельными микросервисами. В таком случае при рассмотрении данной архитектуры она является наиболее эффективной при начале разработки приложения или ведения разработки в маленьких командах.

Если затрагивать гибкость, то монолитную архитектуру трудно назвать таковой. В монолитную архитектуру трудно внести какие-либо дополнительные изменения и функционал, поскольку каждое изменение будет затрагивать все модули приложения.

Далее рассмотрим расширяемость монолитной архитектуры. Как было сказано выше в пункте о её гибкости, добавление нового функционала в приложение параллельно основному не затрагивая его, по своей сути невозможно. В монолитной архитектуре все элементы приложения связаны между собой и поэтому нет никакой перспективы в попытках расширять монолитную архитектуру приложений.

С точки зрения масштабируемости монолитная архитектура является одним из не самых лучших решений. С монолитным кодом способен работать только тот разработчик, который был у «истоков» его написания. Внедрение новых специалистов в разработку является одной из наиболее трудных задач в ходе эксплуатации приложения с монолитной архитектурой. Например, в случае если ваш ключевой разработчик ушел из команды это сильно пошатнет целостность архитектуры приложения и её функциональные возможности.

Тестирование является одним из достоинств монолитной архитектуры. В частности, сквозные тесты, которые легче всего выполнять именно в монолитной архитектуре.

Возможности повторного использования в монолитной архитектуре крайне низкие, так как при написании кода нет полноценно выделенных модулей приложения. В свою очередь использование кода целиком имеет очень низкие шансы на полноценное внедрение в другое приложение.

В заключение, с точки зрения структурированности кода в монолитной архитектуре трудно сделать однозначный вывод. Поскольку внутри каждого приложения имеется своя команда разработчика со своим подходом к реализации приложений. Однако в случае монолитного приложения реализация разграниченной кода с точки зрения модулей и функционала не такая четкая как в других архитектурах, например в микросервисной.

Исходя из всего выше сказанного, можно с уверенностью сказать, что монолитная архитектура имеет свои достоинства и свои недостатки. В начале истории архитектуры приложений и разработки ПО данная архитектура была бесспорным фаворитом, однако со временем, при росте масштаба разработок во многих организациях принцип монолитности был смещен принципом «разделяй и властвуй», а на смену монолитных приложений пришли микросервисные.

Список используемых источников

1. Хамбл Джез, Уиллис Джон, Дебуа Патрик, Ким Джен Руководство по DevOps. Как добиться гибкости, надежности и безопасности мирового уровня в технологических компаниях : руководство / Хамбл Джез, Уиллис

Джон, Дебуа Патрик, Ким Джен; Москва : Манн, Иванов и Фербер, 2018. - 512 с. Электронное издание - ISBN 978-5-00100-750-0. - Текст : электронный.

2. Мартин Фаулер Шаблон корпоративных приложений: учебное пособие / Мартин Фаулер, Дейвид Райс; Москва : Вильямс, 2010. - 539 с. - ISBN 9785845916112, 5845916119. – Текст : непосредственный.
3. Все о микросервисах – URL: https://habr.com/ru/company/raiffeisenbank/blog/346380/ (дата обращения 01.11.2022) Режим доступа: Агрегатор статей habr. Текст: электронный.

List of sources used

1. Humble Jez, Willis John, Debois Patrick, Kim Jen DevOps Guide. How to achieve world-class flexibility, reliability and security in technology companies: a guide / Humble Jez, Willis John, Debois Patrick, Kim Jen; Moscow : Mann, Ivanov and Ferber, 2018. - 512 p. Electronic edition - ISBN 978-5-00100-750-0. - Text : electronic.
2. Martin Fowler Corporate Application Templates: Textbook / Martin Fowler, David Rice; Moscow: Williams, 2010. - 539 p. - ISBN 9785845916112, 5845916119. – Text : direct.
3. All about microservices – URL: https://habr.com/ru/company/raiffeisenbank/blog/346380 / (accessed 01.11.2022) Access mode: Article Aggregator habr. Text: electronic.

Для цитирования: Бежик А.А., Мажей Я.В. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МИКРО-СЕРВИСНОЙ И МОНОЛИТНОЙ АРХИТЕКТУРЫ// Научный сетевой журнал «Столыпинский вестник» №9/2022