ПРИМЕНЕНИЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ В УПРАВЛЕНИИ РИСКАМИ

П. А. Родичкин, В. М. Соловьев

Саратовский национальный исследовательский государственный университет им. Н. Г. Чернышевского, Россия
Email: MrNyanKat@yandex.ru, svm@sgu.ru

В последние годы все популярнее становится микросервисная архитектура. Данная статья посвящена рассмотрению микросервисов, как явления, описаны их основные преимущества и недостатки в сравнении с монолитным подходом к проектирования программного обеспечения. Также микросервисы будут рассмотрены с точки зрения ведения бизнеса, рисков, которые они несут, и возможных путей решения.

APPLICATION OF MICROSERVICE ARCHITECTURE IN RISK MANAGEMENT

P. A. Rodichkin, V. M. Solovyev

In recent years, microservice architecture has become increasingly popular. This article is devoted to the consideration of microservices as a phenomenon, describing their main advantages and disadvantages in comparison with the monolithic approach to software design. Microservices will also be considered from the point of view of doing business, the risks they pose and possible solutions.

Микросервисы [1] прошли долгий путь за более чем десять лет своего существования. Упоминания об этой архитектуре можно встретить в различной литературе начиная с примерно 2000-х годов, но годом «рождения» микросервисов считается 2012 год, когда Джеймс Левис (James Lewis) представил доклад «Java, the Unix Way» на конференции 33d Degree в Кракове. С тех пор микросервисы развивались незаметно, не привлекая особого внимания, но за последние несколько лет ситуация кардинально изменилась.

Микросервисы [2] стали важной темой в IT, привлекая внимание общественности, и технологии в этой области развиваются стремительно, постоянно появляются новые идеи [3].

Чтобы понять, почему возник такой интерес к этой архитектуре, необходимо рассмотреть предыдущий подход к программированию и изучить его проблемы. Монолитный подход, также известный как «монолит» [4], является антиподом микросервисной архитектуры. В монолите все компоненты программного обеспечения находятся в одном месте, представляя собой единое целое. Это сопровождается множеством проблем, которые становятся все более весомыми с увеличением размера программного продукта.

Проблемы создания монолитных приложений включают ограниченность в выборе средств, что отражается в выборе языков программирования и средств внутри каждого языка. Это может затруднить выбор компонентов для сложных приложений, предназначенных для решения множества задач.

Кроме того, сложная структура монолитных приложений может создать трудности при их проектировании, особенно с учетом выбранных технологий и различных частей программного обеспечения.

Проблемы поддержки и дальнейшего развития монолитных приложений включают экспоненциально растущую сложность обслуживания, связанную с увеличением количества переменных, требующих внимания. Это усложняет структуру проекта и делает дальнейшую разработку более сложной.

Кроме того, низкая отказоустойчивость монолитных приложений, вызванная сильной зависимостью компонентов друг от друга и их нахождением в одном месте, может привести к недостаточной надежности программ.

Высокая стоимость разработки и поддержки монолитных приложений обусловлена всеми вышеперечисленными проблемами, а также высокой ценой на квалифицированных специалистов, способных на грамотное проектирование и разработку необходимых компонентов.

Микросервисный подход предлагает неограниченный выбор языков программирования и средств разработки, поскольку каждый компонент взаимодействует с другими через специальные каналы. Этот подход обеспечивает более простую структуру, которая не усложняется с увеличением объемов приложения – достаточно просто добавить новый необходимый компонент по имеющейся схеме.

Он также обеспечивает более высокую надежность, поскольку отказ одного сервиса не приведет к остановке всего приложения.

Кроме того, микросервисный подход имеет более низкую цену на разработку и поддержку, так как он в настоящее время пользуется большой популярностью, и найти подходящих специалистов не составляет особой проблемы. Все это приводит к тому, что бизнес проявляет интерес к микросервисным технологиям.

Однако, несмотря на все преимущества микросервисного подхода, следует учитывать его недостатки.

Во-первых, несмотря на отсутствие ограничений на выбор языков, коммуникация между отдельными частями приложения становится сложнее, что приводит к дополнительным трудностям в проектировании.

Во-вторых, с ростом проекта и добавлением новых сервисов в уже существующий проект, поддержка и разработка такого программного обеспечения становятся все сложнее.

В-третьих, стоит учитывать финансовые аспекты. Хотя микросервисный подход может быть выгодным по сравнению с монолитным подходом, при сложной структуре проекта и его постепенном расширении затраты на квалифицированный персонал, серверы и другое оборудование могут перекрыть преимущества микросервисов.

Анализируя вышеуказанные недостатки, важно отметить, что они все объединены одним общим фактом: все это – проблемы/недостатки плохого планирования.

Из этого следует вывод, что наиболее непродуктивным решением со стороны бизнеса будет бездумное следование трендам и инвестиции в микросервисную разработку без детального анализа ситуации. Это повлечет за собой финансовые потери, которые будут продолжать расти, если приложение останется действующим. Следовательно, необходимо подходить к микросервисной разработке с глубоким пониманием процессов, а также соблюдением некоторых правил, таких как:

Максимально вложить средства в аналитику и планирование будущего проекта. Квалифицированные специалисты помогут разработать детальный план будущего ПО, включая его дальнейшее развитие, что позволит предотвратить финансовые проблемы в будущем.

Также не следует окончательно отбрасывать монолитный подход [5]. На сравнительно небольшом временном отрезке монолит куда проще и дешевле в проектировании. Поэтому самым подходящим вариантом будет создание монолитного прототипа, так называемого MVP (0) (Minimal Viable Product – минимально жизнеспособный продукт), который позволит на конкретном примере оценить целесообразность дальнейших инвестиций в проект и скорректировать последующие действия по его развитию, и лишь затем переходить на микросервисы.

Помимо вышесказанного, важна оценка необходимости микросервисного подхода на каждом этапе проекта. Четкое понимание преимуществ и недостатков микросервисной архитектуры позволит избежать таких необдуманных решений, как, к примеру, перенос монолита на новую архитектуру.

Не смотря на описанные выше моменты, касающиеся данного вопроса, далеко не всегда смена подхода имеет смысл – например, когда проект на текущий момент функционирует, а его жизненный цикл находится в завершающей стадии; или же в случае, когда дальнейшее расширение проекта не планируется, переход на микросервисы повлечет за собой лишь лишние траты, временные потери и так далее.

Микросервисы, являясь, на данный момент, одним из самых перспективных направлений в разработке ПО, стали крайне привлекательным объектом для бизнеса. Но, вместе со всеми преимуществами, микросервисный подход несет в себе и большое количество недостатков, а также рисков, в том числе, и финансовых.

Целью данной статьи являлась попытка всесторонне изучить микросервисы, включая различные их проблемы, а также постараться выработать некоторую абстрактную стратегию, которая позволила бы эффективнее подойти к микросервисам с точки зрения управления проектом в целом, и связанных с этим финансовых рисков.

СПИСОК ЛИТЕРАТУРЫ

1. *Ричардсон К*. Микросервисы. Паттерны разработки и рефакторинга / Книга : Изд-во Питер, 2020. 544 с.

2. *Lewis J*. Microservices. [Электронный ресурс]. URL: https://tinyurl.com/4cv75ykj (дата

обращения: 01.10.2023).

3. *Кочер П. С.* Микросервисы и контейнеры Docker / Книга : Изд-во ДМК-Пресс, 2019. 240 с.

4. *Ньюмен С.* От молонита к микросервисам // БХВ-Петербург, 2021. 272 с.

5. *Dehghani Z.* How to break a Monolith into Microservices. [Электронный ресурс]. URL: https://tinyurl.com/2bcessdv (дата обращения: 03.10.2023).