УДК 004.41

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МОНОЛИТНОЙ И МИКРОСЕРВИСНОЙ АРХИТЕКТУР ИНФОРМАЦИОННЫХ СИСТЕМ

Назарков Д.А.,
студент 2 курса магистратуры
Московский государственный технический университет им. Н.Э. Баумана
Россия, г. Москва

Телышев А.В.,
студент 2 курса магистратуры
Московский государственный технический университет им. Н.Э. Баумана
Россия, г. Москва

Аннотация

В данной статье рассмотрены монолитный и микросервисный подходы к проектированию архитектур информационных систем, проведен сравнительный анализ по ряду критериев.

Ключевые слова: монолитная архитектура, микросервисная архитектура, быстродействие, масштабируемость, надежность, сложность реализации.

COMPARATIVE ANALYSIS OF MONOLITHIC AND MICROSERVICE ARCHITECTURES OF INFORMATION SYSTEMS

Nazarkov. D.A.,
master student
Bauman Moscow State Technical University
Russian Federation, Moscow

Telyshev A.V.,
master student
Bauman Moscow State Technical University
Russian Federation, Moscow

Annotation
In this article monolithic and microservice architecture approaches of information systems architectures are considered, a comparative analysis is carried out according to a number of criteria.

Keywords: monolithic architecture, microservice architecture, performance, scalability, reliability, complexity of implementation.

В настоящее время наблюдается значительный рост числа электронных устройств, имеющих выход в сеть Интернет, данный рост влечет за собой увеличение нагрузки на веб-сервисы, возникают различные проблемы, связанные с их масштабированием [1]. В связи с этими проблемами программисты вынуждены уделять больше времени разработке архитектур своих систем, подстраивая их под требования.

Программную систему можно назвать «монолитной», если весь её код развертывается и запускается в рамках одного процесса на одном узле. Типичным примером системы, построенной по монолитной архитектуре, может являться простое веб-приложение. Такое веб-приложение в общем случае с точки зрения инфраструктуры представляет собой одну машину, на которой работает процесс, принимающий, обрабатывающий клиентские запросы, и при необходимости обращающийся к базе данных системы. По мере роста нагрузки на приложение, время отклика на действия пользователя может существенно снижаться в следствие очевидной причины: приемом, обработкой, запросами к БД занимается один процесс на одной машине.

Микросервисная архитектура – это сервис-ориентированный подход к созданию приложения, подразумевающий отказ от единой, монолитной структуры. То есть вместо того, чтобы исполнять все ограниченные контексты приложения на сервере с помощью внутрипроцессных взаимодействий, система разбивается на набор микросервисов, каждый из которых решает одну или насколько задач в ограниченном контексте. Микросервисы – это небольшие, автономные, совместно работающие сервисы. При этом микросервисы должны быть легко заменяемыми и масштабируемыми [2][3]. Микросервисы в общем случае физически могут работать на разных машинах и взаимодействуют друг с другом посредством различных протоколов, например HTTP/REST или AMQP [4].

Для сравнения монолитной и микросервисной архитектур была разработана и реализована тестовая система в соответствующих архитектурных стилях. В целях демонстрации различий по выбранным критериям между реализациями системы, было принято решение, что система будет представлять собой веб-приложение, способное принимать и обрабатывать пользовательские запросы. Ниже на рисунке 1 представлена общая схема работы системы.

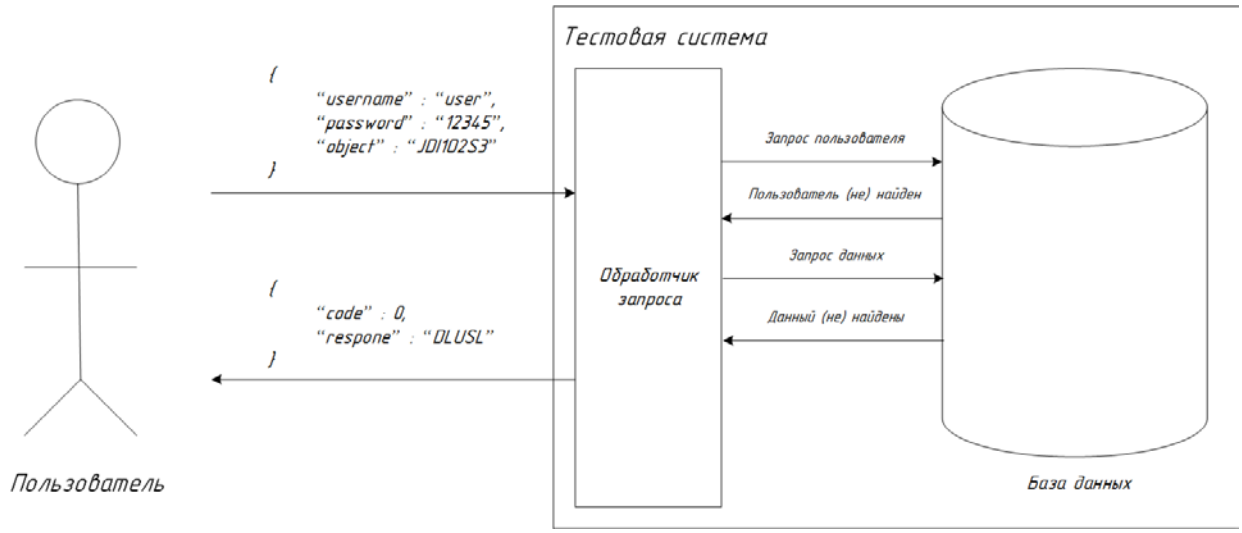

Рисунок 1 – Общая структурная схема тестовой системы

Если рассматривать общие характеристики любых программных систем, не углубляясь в детали функционирования, то можно выделить следующие

наиболее важные характеристики, которые можно рассматривать в качестве критериев для сравнения архитектурных подходов:

Сложность реализации – трудозатраты при построении программной и физической инфраструктуры системы. Для того, чтобы запустить монолит необходимо и достаточно подготовить одну машину, инфраструктуру для неё и запустить само приложение. В отличии от монолитных систем, микросервисные системы в общем случае работают на нескольких машинах. Соответственно для того, чтобы запустить микросервисную систему, во-первых, необходимо выполнить все действия для запуска монолита столько раз, сколько физических узлов есть в системе. Во-вторых, для работы микросервисной системы требуется наладить взаимодействие между микросервисами. В-третьих, микросервисная система в общем случае является гетерогенной, то есть каждый микросервис может быть разработан с использованием различных технологий, что также усложняет реализацию. Исходя из вышеперечисленного, можно сделать вывод, что микросервисная система значительно сложнее в реализации, чем монолит, так как её разработка и дальнейшее развертывание влекут за собой дополнительные трудности.

Быстродействие – свойство системы, характеризующее временной промежуток между запросом пользователя и получением результата. Для того, чтобы оценить соотношение быстродействий монолитной и микросервисной систем, проведем соответствующее тестирование, по завершению которого получим число запросов, обрабатываемых в секунду. По окончанию тестирования были получены результаты, приведенные в таблице 1. По результатам видно, что быстродействие приложения, построенного по монолитной архитектуре, почти в 2 раза выше микросервисного в следствие отсутствия временных потерь на передачу данных между микросервисами по сети.

Таблица 1 – Результаты нагрузочного тестирования монолитной и микросервисной систем

	Монолит	**Микросервисы**
Затраченное время (с)	61.8476	122.5342
RPS	161.6875	81.6098

Масштабируемость – способность системы справляться с ростом нагрузки путем увеличения вычислительных мощностей. Единственный возможный путь увеличения производительности монолита, не меняя его структуры – это вертикальное масштабирование, так как монолитная система может быть запущена только в рамках одного процесса на одной физической машине. Микросервисная система в вопросах масштабируемости имеет значительное преимущество по сравнению с монолитами, так как к ней применимы оба типа масштабирования: вертикальное и горизонтальное. По аналогии с монолитом путем вертикального масштабирования можно легко увеличить производительность отдельно взятых микросервисов. Микросервисная архитектура благодаря своей структуре уже изначально предполагает горизонтальную масштабируемость.

Надежность – способность системы выполнять заданные функции, сохранять характеристики в установленных пределах в процессе эксплуатации. Соответственно, если возникает ошибка в программном коде монолита, приводящая к аварийному завершению, то система целиком приходит в состояние неработоспособности. В отличие от монолита, микросервисная система при возникновении ошибки в одном из микросервисов и выходе его из строя в общем случае теряет лишь функциональность, связанную с ним.

Сложность поддержки – трудозатраты, направленные на расширение функциональных возможностей системы или устранение ошибок. В течение жизненного цикла продукта, как правило, возникает необходимость расширить

или доработать его функциональный состав. Сложность добавления новых функций в монолит напрямую зависит от его размеров и качества его исходного кода. Если монолит сравнительно небольшой, то его расширение – простая задача. Однако, если монолит большой, то вероятность появления скрытых связей между компонентами, которые не позволяют безболезненно добавить новый функционал, увеличивается. Микросервисная система в этом вопросе обладает явным преимуществом, так как микросервис выполняет задачи в рамках одного контекста и является небольшим. Преимуществом микросервисов при добавлении новых возможностей становится взаимодействие между ними через четко определенные интерфейсы, следовательно, реализация функций микросервиса скрыта от потребителей, и её можно изменять.

Подводя итог, следует отметить, что выбирать и строить архитектуру необходимо прежде всего исходя из требований, предъявляемых к ней. Если система на этапе планирования в течение всего своего жизненного цикла представляется небольшой, то строить ее по микросервисной архитектуре нецелесообразно, так как она значительно сложнее в реализации и требует гораздо больших трудозатрат. Если систему не планируется расширять или масштабировать горизонтально, то правильным решением будет разрабатывать её в монолитном стиле, так как разработчик в этом случае получает все преимущества, которые может дать эта архитектура. Микросервисная архитектура призвана решить проблемы больших или очень больших систем. Если строить небольшую систему по микросервисной архитектуре, то все её преимущества нивелируются сложностью разработки, поддержки и управления. С основными преимуществами микросервисной архитектуры разработчики сталкиваются уже на этапе поддержки продукта. Систему легко расширять, масштабировать, устранять дефекты.

Библиографический список:

1. Internet users [Электронный ресурс]. — Режим доступа — URL: http://www.internetlivestats.com/internet-users/ (Дата обращения 30.04.2019).

2. S. Newman Building Microservices. Изд-во O'Reilly, 2016. 304 с.

3. I. Nadareishvili, R. Mitra. M. McLarty, M. Amundsen Microservice Architecture. Изд-во O'Reilly, 2017. 207 с.

4. Pattern: Microservice Architecture [Электронный ресурс]-. Режим доступа: http://microservices.io/patterns/microservices.html - (Дата обращения 30.04.2019)

Оригинальность 97%