DOI: 10.58168/CISMP2024_649-652
УДК 004.9

ПРЕИМУЩЕСТВА ВНЕДРЕНИЯ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

О.В. Оксюта, В.А. Бойченко

ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова»

Аннотация. Статья анализирует преимущества микросервисной архитектуры, такие как модульность, масштабируемость, независимость компонентов и ускорение разработки. Рассматривается сравнение с монолитными системами, подчеркивая гибкость и улучшенную отказоустойчивость микросервисов в современных бизнес-приложениях.

Ключевые слова: микросервис, монолит, архитектура, ускорение разработки, компонент.

ADVANTAGES OF IMPLEMENTING MICROSERVICE ARCHITECTURE

O.V. Oksuyta, V.A. Boychenko

Voronezh State University of Forestry and Technologies named after G.F. Morozov

Abstract. The paper analyzes the advantages of microservice architecture such as modularity, scalability, component independence, and development acceleration. A comparison with monolithic systems is discussed, emphasizing the flexibility and improved fault tolerance of microservices in modern business applications.

Keywords: microservice, monolith, architecture, development acceleration, component.

Введение

Современные технологии требуют гибких и масштабируемых решений для разработки программного обеспечения, особенно в условиях быстро меняющихся бизнес-процессов. Традиционные монолитные архитектуры, несмотря на свою надежность, сталкиваются с трудностями при расширении и поддержке сложных систем. Микросервисная архитектура, напротив, предлагает более модульный и адаптивный подход, при котором каждая функция разрабатывается как независимый сервис. Это позволяет повысить скорость разработки, упростить развертывание, улучшить масштабируемость и гибкость системы. В данной статье рассмотрены ключевые преимущества внедрения микросервисной архитектуры и ее влияние на эффективность разработки и эксплуатацию современных программных решений.

Архитектурные различия

Принятие решения по архитектуре приложения должно основываться на объективных данных таких как: каков будет будущий размер приложения, с каким объемом данных ему предстоит работать, возможность его расширения и кратного увеличения команды разработчиков, занимающихся разработкой данного продукта.

Монолитная архитектура предполагает, что все функции и модули приложения объединены в одно целое. Компоненты тесно связаны между собой, и изменение одного из них может повлиять на работу всей системы. Это делает монолитные приложения сложными в поддержке, масштабировании и обновлении. Любое обновление требует переработки и повторного развертывания всего приложения, что увеличивает время и риски внесения изменений. [1]

Микросервисная архитектура, напротив, разделяет систему на набор независимых сервисов, каждый из которых отвечает за конкретную бизнес-функцию. Эти микросервисы могут разрабатываться, развертываться и масштабироваться независимо друг от друга. Они взаимодействуют через легковесные протоколы, такие как HTTP или AMQP, обычно через API. Каждый сервис имеет свою собственную базу данных и логику, что позволяет снижать взаимозависимость и устранять "узкие места" в системе.

Такая структура способствует упрощению управления сложными системами, ускоряет процесс обновления отдельных частей системы без необходимости полного развертывания. Это также увеличивает гибкость в выборе технологий: каждая команда может использовать разные языки программирования или базы данных для своих микросервисов, что делает систему более адаптивной к изменениям и новым требованиям. [2]

На основании вышеперечисленных описаний двух архитектур можно резюмировать некоторые очевидные случаи, когда следует выбирать ту или иную архитектуру. Например, микросервисная архитектура подходит для приложений, которым предъявляются требования по масштабируемости и гибкости, сложность которых слишком велика для того, чтобы их разрабатывала только одна команда. Монолитная же архитектура идеально подходит для малого и среднего размера проектов: в таких случаях не нужно задумываться над тем, каким проект может стать, если его потребуется значительно расширить или масштабировать. Обычно такие проекты самодостаточны или являются частью какой-то большой системы, поэтому разделение их на своеобразные модули, как в микросервисной архитектуре, привносит лишь излишнюю сложность проекта, временные издержки на взаимодействия разделенных источников данных и их обработчиков и необязательную сложность разработки для команды.

Разумеется, в «боевой» разработке могут быть случаи, когда выбор архитектуры не является настолько легким решением. Более того, сейчас существуют инструменты разработки, которые упрощают некоторые процессы, которые представляют трудность для команд разработки. Например, для управления системой на микросервисной архитектуре используется связка

Docker + Kubernetes, которая избавляет от трудностей, связанных с управлением системой как единым целым.

Преимущества перехода существующих монолитов на микросервисы

У каждого большого монолита со временем наступает точка невозврата, когда дальнейшее расширение в рамках простой монолитной архитектуры сулит значительное усложнение проекта, возрастающие издержки ресурсов как команды, так и инфрастуктуры.

По мере роста монолитного приложения разработка новых функций становится всё более сложной и медленной. Каждое изменение может затронуть другие части системы, что увеличивает время на тестирование и внедрение новых функциональностей. Микросервисы позволяют разрабатывать и внедрять изменения в отдельных компонентах независимо друг от друга, что ускоряет процесс и снижает риски ошибок. [3]

Монолитная архитектура ограничивает возможность использования разных технологий для разных частей системы. Все компоненты вынуждены работать на одном технологическом стеке. Микросервисы, напротив, позволяют использовать различные языки программирования, базы данных и инструменты для разных сервисов, что дает больше гибкости в выборе наилучших решений для каждой задачи.

Если приложение постоянно растет и обрастает новыми бизнес-функциями, монолитная архитектура может стать перегруженной. В результате поддержка и модификация системы становятся более трудоемкими. Микросервисы помогают структурировать бизнес-логику по отдельным сервисам, каждый из которых отвечает за свою задачу. Это делает систему более управляемой и легко расширяемой. [4]

По мере того как монолитное приложение растет, оно может становиться сложно поддерживаемым, особенно если оно разрабатывалось долгие годы разными командами. В таком случае кодовая база может быть перенасыщена техническим долгом, что затрудняет её изменение и адаптацию под новые требования. Разбиение системы на микросервисы позволяет улучшить организацию кода и избавиться от устаревших или плохо поддерживаемых участков. [5]

Выводы

Внедрение микросервисной архитектуры стало одной из ключевых тенденций в разработке современных программных систем. Микросервисы позволяют преодолеть ограничения монолитной архитектуры, предлагая более гибкий и масштабируемый подход, особенно для сложных и крупных приложений. Разделение приложения на независимые сервисы позволяет каждой команде разрабатывать, тестировать и внедрять изменения автономно, что ускоряет процесс разработки и повышает качество продукта.

Ключевые особенности, преимущества и недостатки микросервисной и монолитной архитектур, разобранные в этой статье, позволяют сделать вывод о том, для каких приложений следует выбирать ту или иную архитектуру.

Если говорить конкретно о микросервисной архитектуре, то в итоге, решение о переходе на микросервисную архитектуру должно приниматься на основе конкретных потребностей и масштабов системы.

Список литературы

1. Митра, Р. Микросервисы. От архитектуры до релиза / Р. Митра, И. Надареишвили // Программная инженерия. – Питер, 2023 – С. 36-38.

2. Understanding Microservices Architecture: A Deep Dive Beyond the Basics. – URL: https://medium.com/@gustavowill/understanding-microservices-architecture-662d1f9fd114 (дата обращения 23.09.2024).

3. Ньюмен, С. От монолита к микросервисам / С. Ньюмен // Основные сведения о микрослужбах. – 2021. – С. 21-29.

4. Microservices vs. monolithic architecture. – URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith (дата обращения 25.09.2024).

5. Monoliths vs. Microservices: Differences, Pros & Cons, and Choosing the Right Architecture. – URL: https://www.cortex.io/post/monoliths-vs-microservices-whats-the-difference (дата обращения 21.09.2024).

6. Юров, А.Н. Разработка автономных программных решений на основе геометрических ядер по созданию цифровых сборочных моделей станочных приспособлений / А.Н. Юров, В.И. Анциферова // Моделирование систем и процессов. – 2020. – Т. 13, № 3. – С. 86-93.

References

1. Mitra, R. Microservices. From architecture to release / R. Mitra, I. Nadareishvili // Software Engineering, St. Petersburg, 2023 – pp. 36-38.

2. Understanding Microservices Architecture: A Deep Dive Beyond the Basics. – URL: https://medium.com/@gustavowill/understanding-microservices-architecture-662d1f9fd114 (accessed 09/23/2024).

3. Newman, S. From monolith to microservices / S. Newman // Basic information about microservices – 2021 – pp. 21-29.

4. Microservices vs. monolithic architecture. – URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith (accessed 09/25/2024).

5. Monoliths vs. Microservices: Differences, Pros & Cons, and Choosing the Right Architecture. – URL: https://www.cortex.io/post/monoliths-vs-microservices-whats-the-difference (accessed 09/21/2024).

6. Yurov, A.N. Development of autonomous software solutions based on geometric cores for the creation of digital assembly models of machine tools / A.N. Yurov, V.I. Antsiferova // Modeling of systems and processes. – 2020. – Vol. 13, No. 3. – pp. 86-93.