УДК 004.27

МИКРОСЕРВИСНАЯ АРХИТЕКТУРА: ПРЕИМУЩЕСТВА И НЕДОСТАТКИ ПРИМЕНЕНИЯ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Мамонтов П.М.,

студент,

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Никитина А.А.,

студент,

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Соколов Н.В.,

старший преподаватель кафедры информатики и информационных технологий

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Аннотация

В данной статье рассматриваются ключевые принципы микросервисной архитектуры, её преимущества и недостатки при разработке информационных систем. Рассмотрены вопросы масштабируемости, отказоустойчивости, взаимодействия через программные интерфейсы, децентрализации информации и автономности компонентов. Проанализированы основные проблемы, возникающие при использовании микросервисного подхода, включая сложность управления, проверки работоспособности и обеспечения защиты. Приведены примеры применения в разных отраслях, включая торговлю, финансы,

стриминговые платформы и здравоохранение. Делается вывод о целесообразности использования микросервисов с учетом специфики проекта.

Ключевые слова: микросервисная архитектура, API, масштабируемость, отказоустойчивость, децентрализация данных, DevOps.

MICROSERVICE ARCHITECTURE: ADVANTAGES AND DISADVANTAGES OF APPLICATION IN INFORMATION SYSTEMS

Mamontov P.M.,
student,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia

Nikitina A.A.,
student,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia

Sokolov N.V.,
Senior Lecturer at the Department of Computer Science and Information Technology
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia

Abstract

This article discusses the key principles of the micro-service architecture, its advantages and disadvantages in the development of information systems. The issues of scalability, fault tolerance, interaction through software interfaces, decentralization of information and autonomy of components are considered. The main problems that arise when using the microservice approach are analyzed, including the complexity of management, health checks, and security. Examples of applications in various industries are given, including trade, finance, streaming platforms, and healthcare. The

conclusion is made about the expediency of using microservices, taking into account the specifics of the project.

Keywords: microservice architecture, API, scalability, fault tolerance, data decentralization, DevOps.

Микросервисная архитектура (MCA) — это новейший способ проектирования программных систем, основанный на разделении приложения на независимые сервисы. Каждый микросервис выполняет ограниченную бизнес-функцию, разрабатывается и развертывается отдельно, что повышает гибкость, устойчивость к сбоям и масштабируемость системы [7].

Для лучшего понимания концепции микросервисной архитектуры приведём сравнительную таблицу с монолитной архитектурой (таблица 1).

Таблица 1. Сравнение монолитной и микросервисной архитектур [4].

Параметр	Монолитная архитектура	Микросервисная архитектура
Структура	Единая программа	Набор независимых сервисов
Разработка	Ведется одной командой	Могут работать несколько команд
Технологии	Единый стек технологий	Возможность выбора технологий для каждого сервиса
Масштабируемость	Масштабируется только целиком	Масштабируются отдельные сервисы
Обновления	Требуется перезапуск всей системы	Локальные обновления без влияния на другие сервисы
Отказоустойчивость	Сбой в одном модуле нарушает работу всей системы	Сбои ограничиваются конкретным микросервисом

Рост популярности MCA связан с развитием облачных вычислений, технологий DevOps и потребностью бизнеса в адаптивных решениях. Многие крупные компании, такие как Netflix, Amazon и Google, используют микросервисный подход для обеспечения высокой надежности и производительности своих систем [5].

Однако наряду с преимуществами этот подход имеет и недостатки, связанные с администрированием сложных распределённых систем.

Целью данного исследования является рассмотрение основных принципов микросервисной архитектуры, её преимуществ и недостатков, а также определение случаев, когда её использование является оправданным.

Микросервисная архитектура стала одним из ключевых подходов в современной разработке программного обеспечения, предоставляя возможность создавать сложные системы, состоящие из множества мелких, независимых компонентов. Основные принципы микросервисной архитектуры направлены на повышение гибкости, устойчивости и масштабируемости системы, а также упрощают процессы разработки и поддержки. Рассмотрим ключевые аспекты этого подхода, которые определяют успех проектов, построенных на основе микросервисов [1;8].

Одним из ключевых принципов МСА является автономность сервисов. Данный принцип заключается в том, что каждый микросервис разрабатывается, тестируется, развертывается и масштабируется обособлено. Это достигается за счёт следующих аспектов:

- **Изоляция бизнес-логики.** Каждый микросервис выполняет отдельную функцию (например, обработка платежей, авторизация пользователей).
- **Независимые обновления.** Для того, чтобы внести изменения в один из сервисов, не требуется перезапускать всю систему.
- **Использование разных технологий.** Микросервисы могут разрабатываться на разных языках программирования, использовать разные базы данных и фреймворки.

Вторым ключевым принципом МСА является коммуникация между микросервисами через API, что позволяет обеспечить обмен данными в едином формате. Наиболее распространены следующие виды:

- **REST API** (на основе HTTP-запросов).

- **gRPC** (более эффективен для бинарных данных).
- **GraphQL** (позволяет клиенту запрашивать только нужные данные).

Микросервисы могут обмениваться информацией синхронно (запрос-ответ) или асинхронно (использование очередей сообщений, например Kafka или RabbitMQ).

Для более глубокого понимания приведём таблицу 2, отражающую ключевые параметры API в микросервисной архитектуре:

Таблица 2. Ключевые параметры API [7].

Параметр	Описание
Типы API	REST, gRPC, GraphQL, WebSocket. Каждый подходит для разных типов задач
Протоколы взаимодействия	HTTP/HTTPS, HTTP/2, TCP, что обеспечивает гибкость для выбора технологии
Форматы данных	JSON, XML, Protocol Buffers (protobuf), Avro. Форматы зависят от протокола
Стандарты безопасности	OAuth 2.0, OpenID Connect, JWT для аутентификации и авторизации
Методы передачи данных	Синхронные (запрос-ответ) и асинхронные (очереди сообщений, событийная модель)
Тестирование	Используются спецификации, такие как OpenAPI/Swagger, для описания API
Документация	Обязательное наличие спецификаций, которые упрощают интеграцию между командами

Следующим принципом MCA является так называемая децентрализация данных, который заключается в том, что каждый микросервис управляет собственными данными, что уменьшает зависимости между сервисами. Это позволяет использовать оптимальные базы данных для каждой задачи. Так, например, для транзакционных данных могут использоваться реляционные базы данных (PostgreSQL, MySQL), для хранения неструктурированных данных NoSQL (MongoDB, Cassandra), а для задач кэширования можно использовать временные хранилища (Redis).

Четвертым принципом микросервисной архитектуры является её масштабируемость и отказоустойчивость. MCA использует горизонтальное масштабирование, увеличивая количество экземпляров отдельных сервисов.

Данный принцип позволяет повысить отказоустойчивость, т.е. сбой одного сервиса не влияет на всю систему. При этом могут применяться такие механизмы как балансировка нагрузки, которая заключается в распределении запросов между доступными сервисами, защита от перегрузки при сбоях (circuit breaker) и автоматическое восстановление сервисов.

Анализ исследований по применению микросервисной архитектуры позволил выделить следующие преимущества и недостатки, представленные в таблице 3.

Таблица 3. Преимущества и недостатки микросервисной архитектуры [2;3].

Преимущества		Недостатки	
Наименование	Описание	Наименование	Описание
Гибкость разработки и развертывания.	Различные команды могут разрабатывать и развертывать микросервисы независимо друг от друга и иметь возможность постепенного обновления системы без остановки её работы.	Усложнение управления системой	Требуется координация множества сервисов. Сложность мониторинга и логирования.
Масштабируемость и отказоустойчивость	Отдельные сервисы могут масштабироваться независимо и отказ одного сервиса не влияет на работу всей системы.	Проблемы с сетевыми взаимодействиями	Высокие накладные расходы на сетевые запросы. Возможны задержки и сбои при передаче данных.
Улучшение управления кодом	Каждый микросервис содержит меньше кода, чем монолитное приложение, что упрощает сопровождение.	Сложность тестирования	Интеграционное тестирование требует запуска множества сервисов одновременно.
Поддержка облачных технологий	Микросервисы хорошо интегрируются с облачными платформами (AWS, Azure, Google Cloud).	Обеспечение безопасности	Каждому сервису требуется механизм аутентификации и авторизации (например, OAuth 2.0, JWT), а также есть высокий риск атак на API.

На сегодняшний день МСА активно используется в различных сферах, где требуется высокая масштабируемость, гибкость и устойчивость систем. Рассмотрим несколько ярких примеров её использования в реальных приложениях в различных областях [1;6;8]:

1) Электронная коммерция - Amazon и eBay используют микросервисы для управления заказами, каталогами товаров, платежами.

2) Стриминговые платформы – платформа Nuum, Rutube масштабирует сервисы обработки видео, рекомендаций, авторизации.

3) Финансовые технологии - PayPal и Revolut используют микросервисы для управления транзакциями, аналитики и обработки данных.

4) Социальные сети – Вконтакте, Одноклассники используют микросервисы для хранения данных профилей, новостных лент, чатов.

5) Здравоохранение - медицинские информационные системы используют микросервисы для управления записями пациентов, обработки видеозвонков и аналитики.

Несмотря на множество преимуществ, микросервисная архитектура подходит не для всех систем и проектов. В ряде ситуаций её применение может быть нецелесообразным или даже вредным. Так, например, МСА нецелесообразна для:

- Малых проектов с ограниченным бюджетом.
- Прототипов, которые не требуют масштабирования.
- Систем с жёсткими требованиями к согласованности данных.

Таким образом микросервисная архитектура обеспечивает гибкость, масштабируемость и отказоустойчивость, что делает её востребованной в современных информационных системах. Однако её внедрение требует значительных ресурсов, сложного управления и глубокого технического понимания. Применение микросервисов оправдано в крупных, высоконагруженных проектах, где преимущества перевешивают затраты на сложность разработки и поддержки.

Библиографический список:

1. Akhmetzyanov, I. I. Using microservice architecture in software development / I. I. Akhmetzyanov, I. N. Urakhchinsky // , 19 мая 2022 года. Vol. Часть III, 2022. – P. 149-153. [Электронный ресурс]. — Режим доступа — URL: https://www.elibrary.ru/item.asp?id=49454674 (Дата обращения 13.01.2025)

2. Бедняк С. Г. Микросервисная архитектура: преимущества и недостатки в разработке информационных систем / С. Г. Бедняк, Д. А. Бугрова // Актуальные проблемы информатики, радиотехники и связи : Материалы XXXI Российской научно-технической конференции, Самара, 01–02 февраля 2024 года. – Самара: Поволжский государственный университет телекоммуникаций и информатики, 2024. – С. 212-214. [Электронный ресурс]. — Режим доступа — URL: https://www.elibrary.ru/item.asp?id=63644677 (Дата обращения 17.01.2025)

3. Белов З. И. Архитектура микросервисов: преимущества, недостатки и применение / З. И. Белов, В. В. Коляда // Информационное общество: современное состояние и перспективы развития : Сборник материалов XV международного форума, Краснодар, 10–14 июля 2023 года. – Краснодар: Кубанский государственный аграрный университет имени И.Т. Трубилина, 2023. – С. 137-139. [Электронный ресурс]. — Режим доступа — URL: https://www.elibrary.ru/item.asp?id=54517870 (Дата обращения 17.01.2025)

4. Егоркин, Е. С. Сравнение монолитной и микросервисной архитектуры в создании современных и удобных веб - приложений / Е. С. Егоркин // Синтез науки и общества в решении глобальных проблем современности : Сборник статей по итогам Международной научно-практической конференции, Тюмень., 30 мая 2024 года. – Стерлитамак: ООО "Агентство международных исследований", 2024. – С. 183-186. [Электронный ресурс]. — Режим доступа — URL: https://www.elibrary.ru/item.asp?id=67334736 (Дата обращения 15.01.2025)

5. Коновалов Г. Г. Анализ применения микросервисной архитектуры при разработке веб-приложений / Г. Г. Коновалов // Тенденции развития науки и образования. – 2023. – № 104-14. – С. 38-41. [Электронный ресурс]. — Режим

доступа — URL: https://www.elibrary.ru/item.asp?id=61103306 (Дата обращения 25.01.2025)

6. Пономаренко В. П. Роль микросервисной архитектуры в современной разработке программного обеспечения / В. П. Пономаренко, Н. И. Белодед // Актуальные проблемы социально-экономического развития современного общества : Материалы V международной научно- практической конференции, Киров, 29 мая 2024 года. – Киров: Кировский государственный медицинский университет, 2024. – С. 538-540. [Электронный ресурс]. — Режим доступа — URL: https://www.elibrary.ru/item.asp?id=67973113 (Дата обращения 20.01.2025)

7. Системный аналитик. Вопросы и ответы для собеседования. [Электронный ресурс]. — Режим доступа — URL: https://system-analyst.tilda.ws/ (Дата обращения 15.01.2025)

8. Фролова Т. В. Микросервисная архитектура как инновационный подход к разработке программного обеспечения / Т. В. Фролова, К. А. Ковалева // Современные стратегии и цифровые трансформации устойчивого развития общества, образования и науки : сборник материалов XIII Международной научно-практической конференции, Москва, 13 ноября 2023 года. – Москва: Алеф, 2023. – С. 117-124. [Электронный ресурс]. — Режим доступа — URL: https://www.elibrary.ru/item.asp?id=59099623 (Дата обращения 16.01.2025)

Оригинальность 80%