И. А. Михневич

# СРАВНЕНИЕ ОСНОВНЫХ СПОСОБОВ ВЗАИМОДЕЙСТВИЯ МИКРОСЕРВИСОВ

*В работе рассматриваются основные подходы взаимодействия микросервисов, определяющие способы обмена данными и координации между сервисами. Такие как REST API, gRPC, GraphQL, брокеры сообщений RabbitMQ и Apache Kafka.*

## Введение

Существует несколько наиболее распространённых способов взаимодействия микросервисов. Разработчиками используются как синхронные подходы — REST API и gRPC и GraphQL, – так и асинхронные механизмы, реализуемые через брокеры сообщений, как RabbitMQ и Apache Kafka.

## I. Подходы к реализации взаимодействия микросервисов

В микросервисных системах взаимодействие обычно строится вокруг двух паттернов: API Gateway, который обеспечивает единую точку входа и централизует маршрутизацию запросов, и Service Registry, хранящий актуальные адреса всех сервисов.

Синхронный подход остаётся самым простым: он работает по модели клиент–сервер, облегчает реализацию последовательных операций и транзакций. Однако он чувствителен к сбоям — недоступность одного сервиса блокирует цепочку вызовов, увеличивает задержки и может приводить к зависаниям всей системы.[2]

Асинхронное взаимодействие решает эти проблемы: сервисы обрабатывают сообщения независимо, не ожидают мгновенного ответа, а данные могут накапливаться в очередях. Это повышает устойчивость и масштабируемость, но усложняет разработку — приходится учитывать порядок и дублирование сообщений, обеспечивать согласованность данных, подтверждение доставки и разбираться со сложной трассировкой.[2]

## II. Способы реализации взаимодействия микросервисов

GraphQL — синхронный подход, где клиент сам выбирает нужные поля. Не устарел, но используется точечно, чаще как API-шлюз. Сложнее реализуется и хуже кешируется. Чаще применяется в API-шлюзах и фронтенд-ориентированных сервисах, не подходит для взаимодействия микросервисов.

HTTP/REST — простой и универсальный способ синхронного взаимодействия по HTTP, удобный для интеграций, но не самый быстрый и может давать лишние данные. Основан на модели request-response с передачей данных в формате XML или JSON.[1]

gRPC — высокопроизводительный бинарный RPC поверх HTTP/2. Даёт низкую задержку и строгие контракты, но менее удобен для отладки и требует более сложной инфраструктуры.[1]

Kafka — брокер собщений, представляющий распределённый лог событий для высоконагруженных асинхронных систем. Отлично подходит для потоковой обработки и event-driven архитектур, но требует сложной инфраструктуры.

RabbitMQ — брокер очередей для асинхронного обмена сообщениями. Хорош для надёжной доставки и разгрузки сервисов, но не рассчитан на огромные потоки данных.

## III. Выводы

REST подходит для простых интеграций и внешних API, когда важна универсальность и читаемость, а требования к производительности умеренные. gRPC целесообразно использовать во внутреннем взаимодействии микросервисов, где критичны скорость, низкая задержка и строгие контракты. GraphQL оправдан в случаях, когда клиенту требуется гибкая выборка данных и агрегация из нескольких источников, но он неэффективен как основной механизм связи микросервисов. RabbitMQ лучше применять там, где важна гарантированная доставка сообщений, надёжность и контролируемая маршрутизация задач. Kafka предпочтительна для высоконагруженных систем, потоковой обработки и событийно-ориентированной архитектуры, обеспечивая масштабируемость и работу с большими объёмами данных.

### *Список литературы*

1. Ричардсон, К. Паттерны разработки и рефакторинга микросервисов / К. Ричардсон. – М.: ДМК Пресс, 2022. – 432 с.
2. adametsderscopfer: Взаимодействие микросервисов между собой [Электронный ресурс] / adametsderscopfer // Habr. – Режим доступа: https://habr.com/ru/articles/844100/. – Дата доступа: 22.03.2026.

*Михневич Иван Александрович*, студент 3 курса ФИТиУ БГУИР, ivanmichnevih2@gmail.com.
*Научный руководитель: Радченко Анастасия Дмитриевна*, ассистент кафедры ВМиП БГУИР, a.radchenko@bsuir