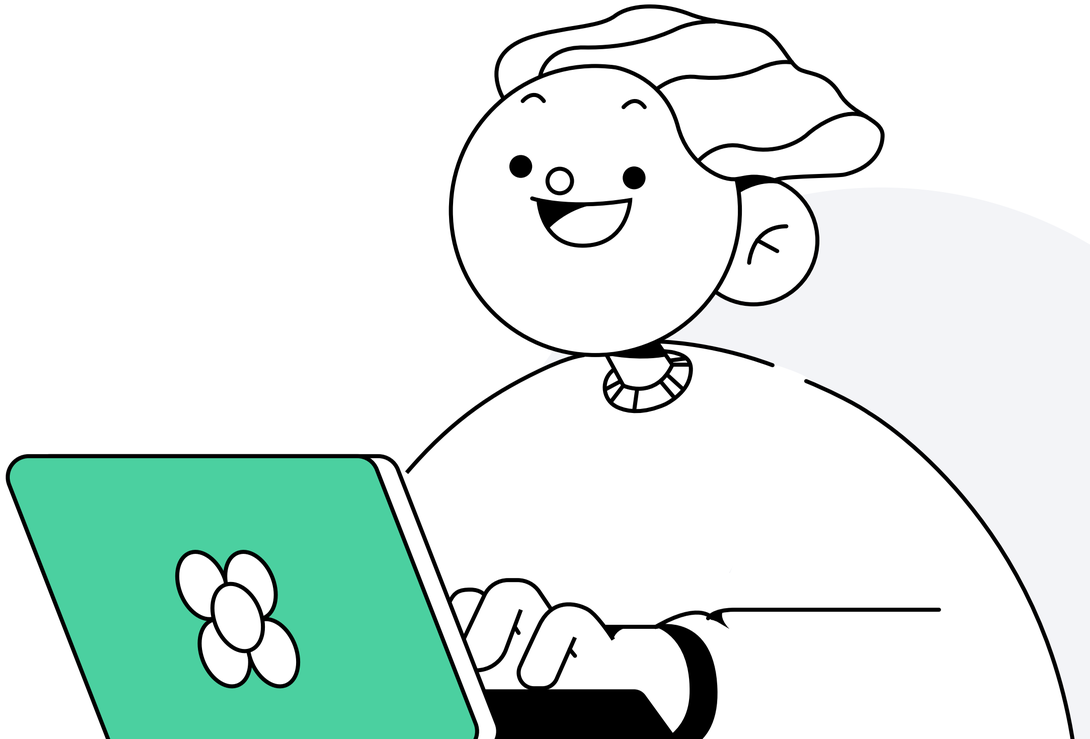
Введение в микросервисы

Михаил Триполитов
Банк Открытие, архитектор решений

План занятия

1. Микросервисы
2. Преимущества
3. Сопутствующие проблемы
4. Антипаттерны
5. Сложные решения

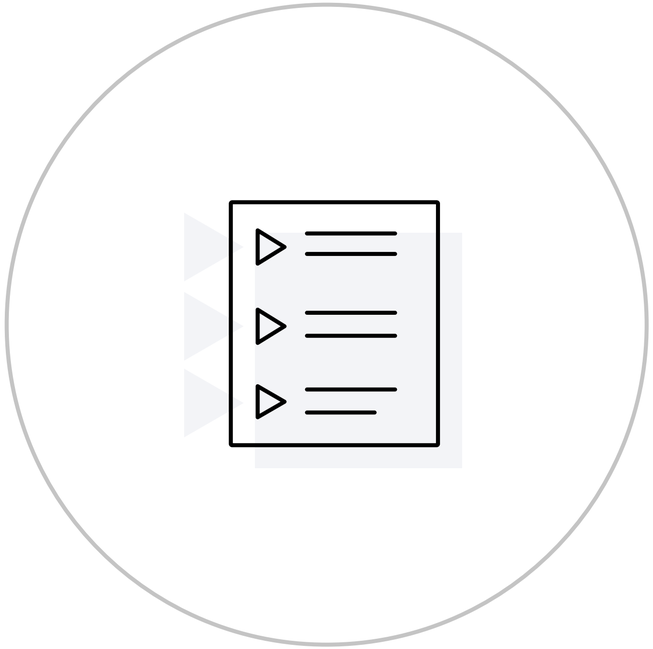


Микросервисы

Михаил Триполитов
Банк Открытие, архитектор решений

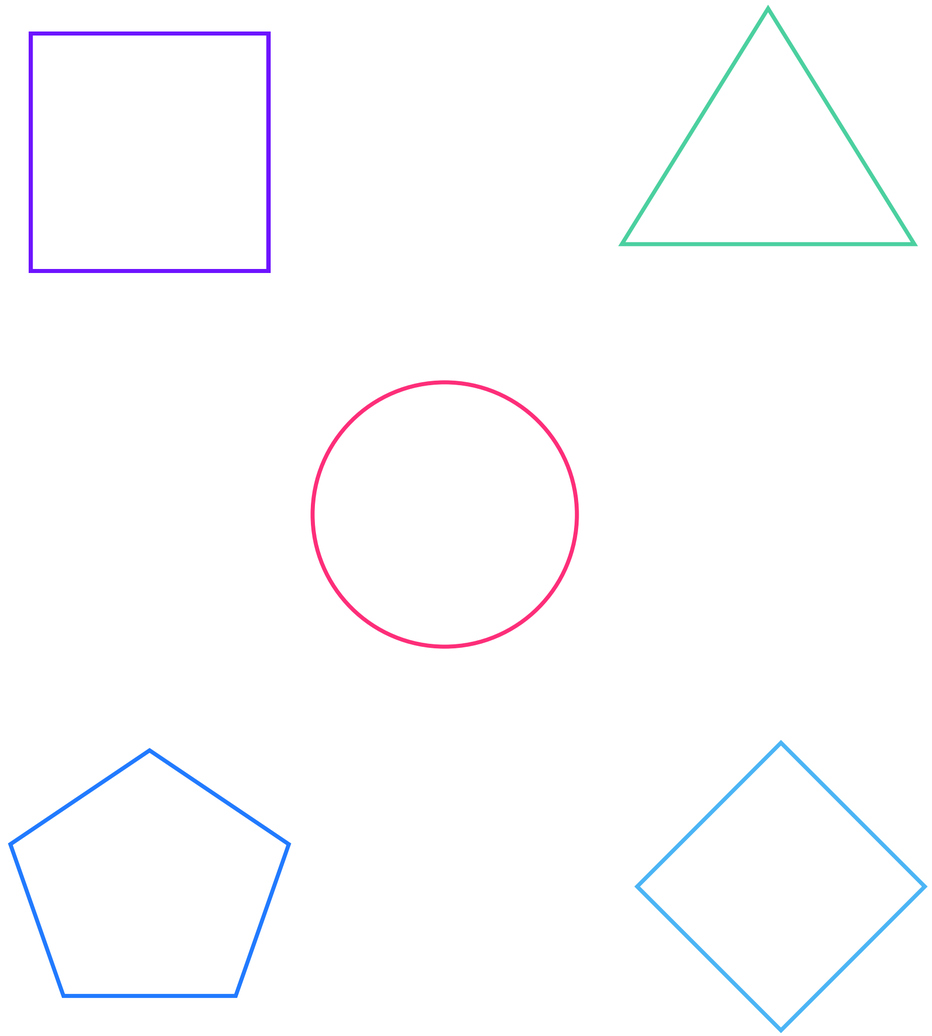
Тема занятия

1. Понятие микросервиса
2. Характеристики микросервисов
3. Цели использования



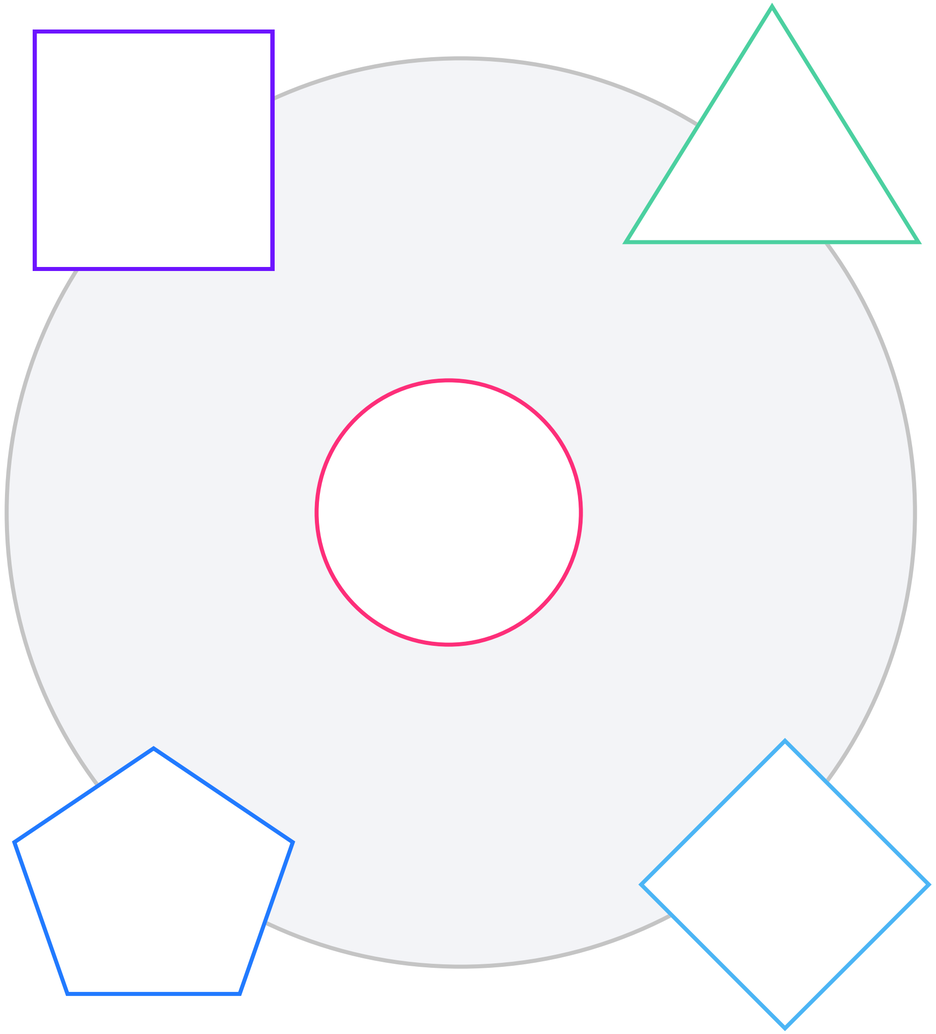
Что такое микросервисы

Микросервисы — это архитектурный подход разделения системы на небольшие автономные сервисы, которые запускаются как отдельные процессы и взаимодействуют, используя API на основе лёгких протоколов, например, HTTP



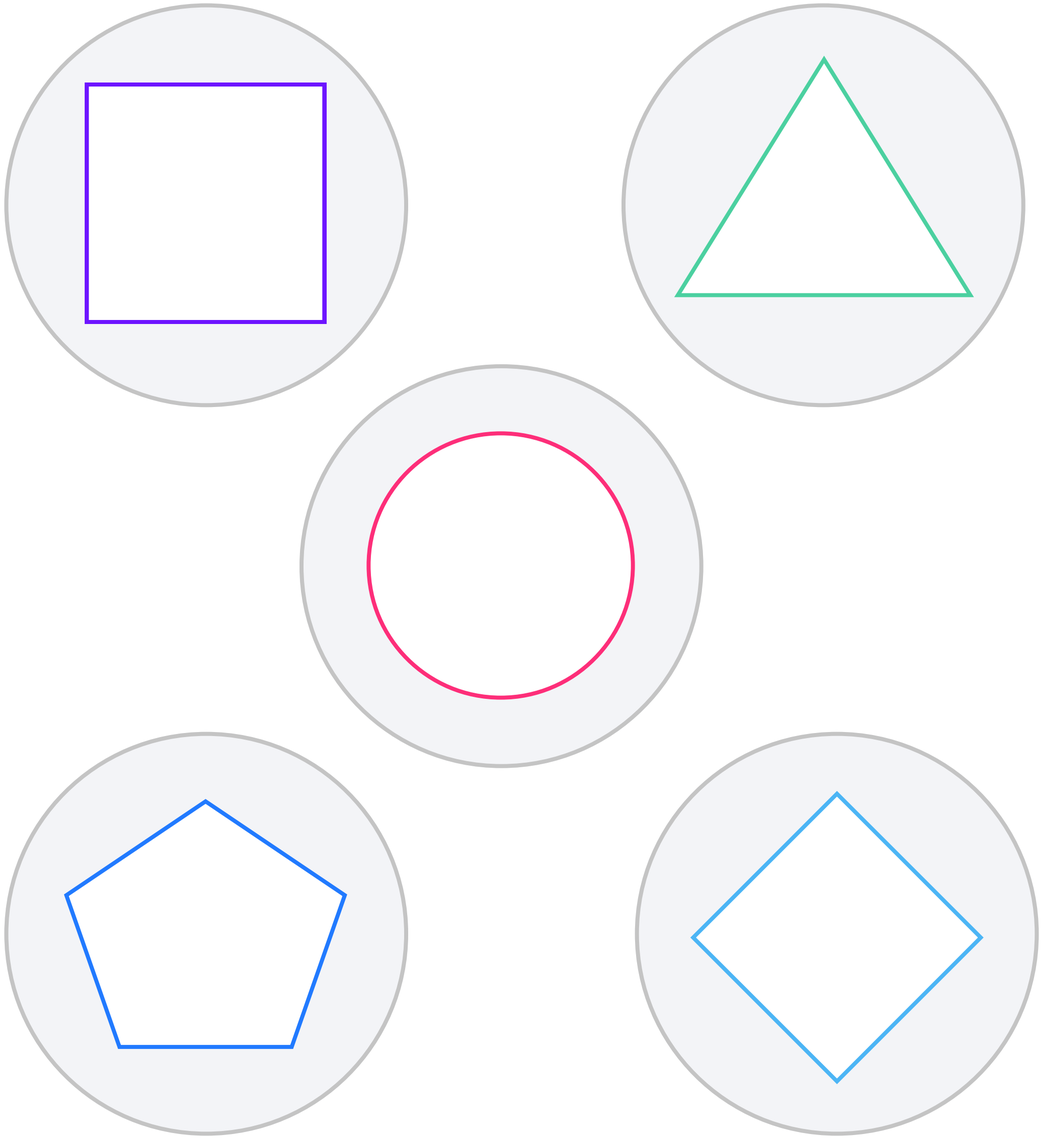
Что такое микросервисы

Монолитное приложение содержит все бизнес-функции в одном процессе



Что такое микросервисы

Микросервисы распределяют бизнес-функции по разным независимым сервисам



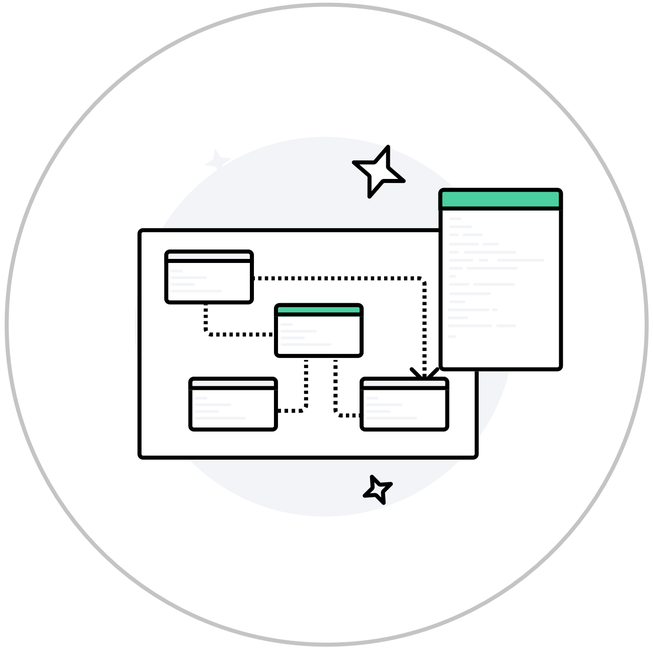
Характеристики микросервисов

1. Слабая связность → команда работает независимо от изменений в других сервисах
2. Независимы при выкладке → нет необходимости координировать выкладку с другими командами
3. Поддерживаемость и тестируемость → быстрая разработка и частые выкладки
4. Наличие кросс-функциональной команды-владельца → сокращает расходы на коммуникации
5. Организация вокруг бизнес-функций → глубокое понимание домена

Зачем использовать

Микросервисная архитектура обладает преимуществами при построении систем:

- с большим количеством интеграций
- с часто меняющейся неоднородной нагрузкой
- обеспечивающих работу различных бизнес-подразделений



Итоги

- **Микросервисы** — предоставление определённому лицу или группе лиц прав на выполнение определённых действий

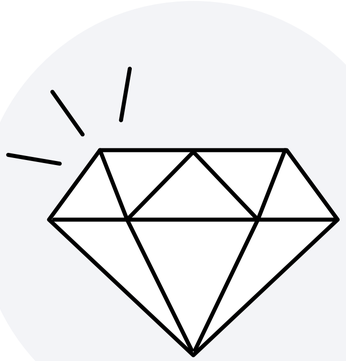
- **Микросервисы** позволяют обеспечить быстрые и частые изменения больших, сложных систем

Итоги

Микросервисы — предоставление определённому лицу или группе лиц прав на выполнение определённых действий

Микросервисы позволяют обеспечить быстрые и частые изменения больших, сложных систем

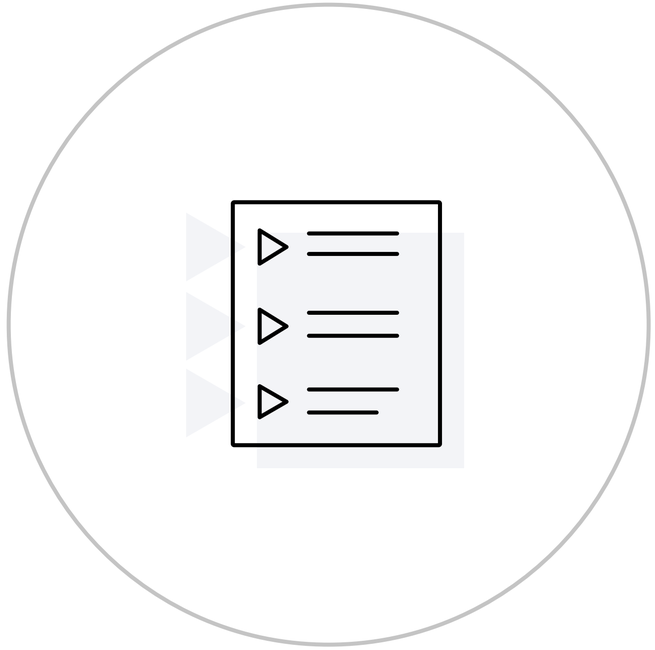
Преимущества



Михаил Триполитов
Банк Открытие, архитектор решений

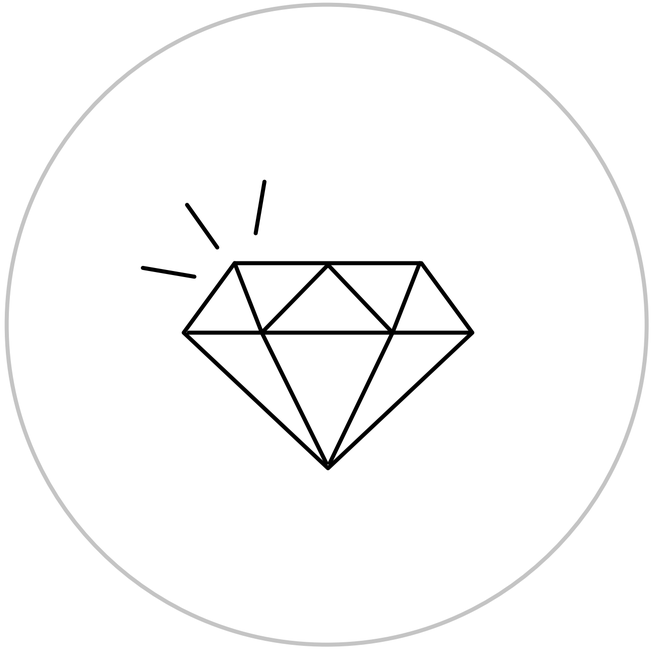
Тема занятия

1. Преимущества разделения системы на сервисы

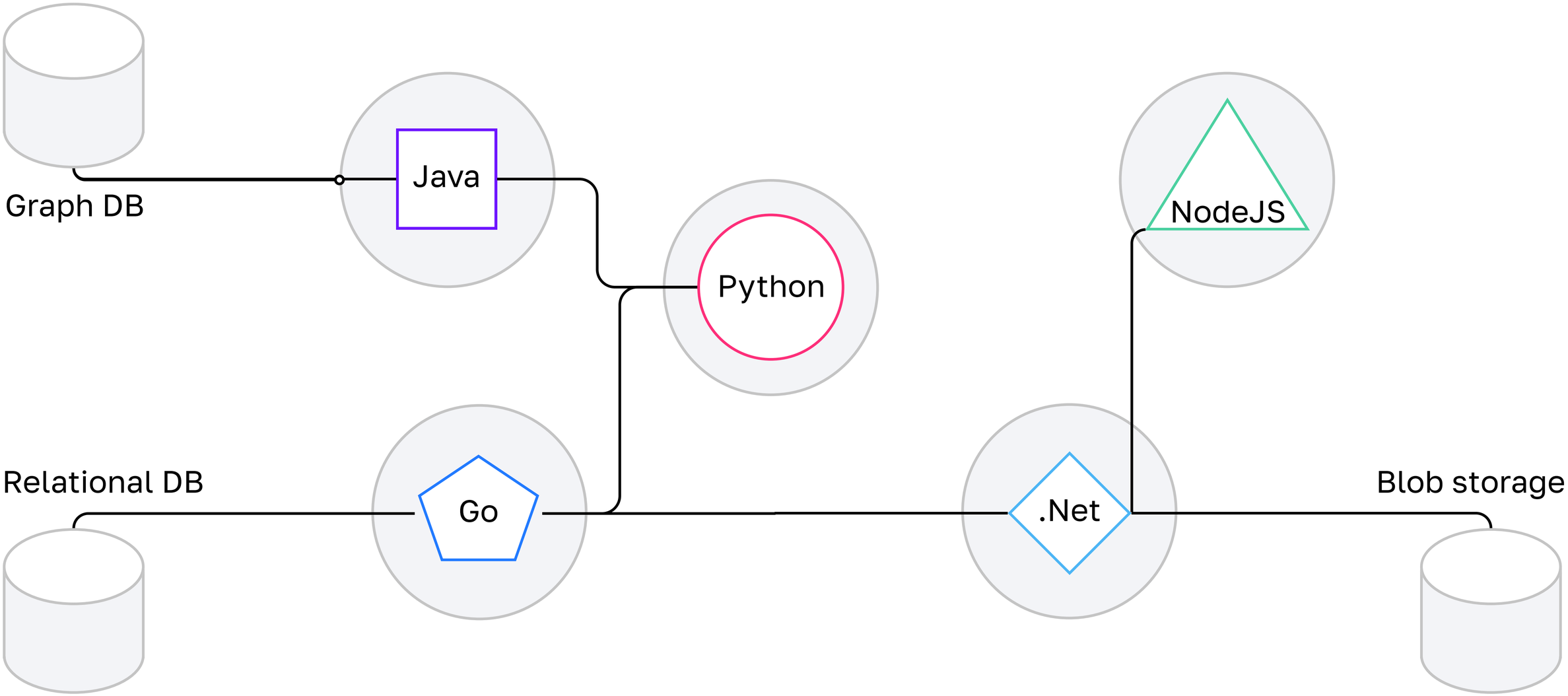


Преимущества

- Возможность использовать разные технологии
- Устойчивость к ошибкам
- Масштабируемость
- Простота развёртывания
- Простота замены
- Отражение структуры организации



Разные технологии

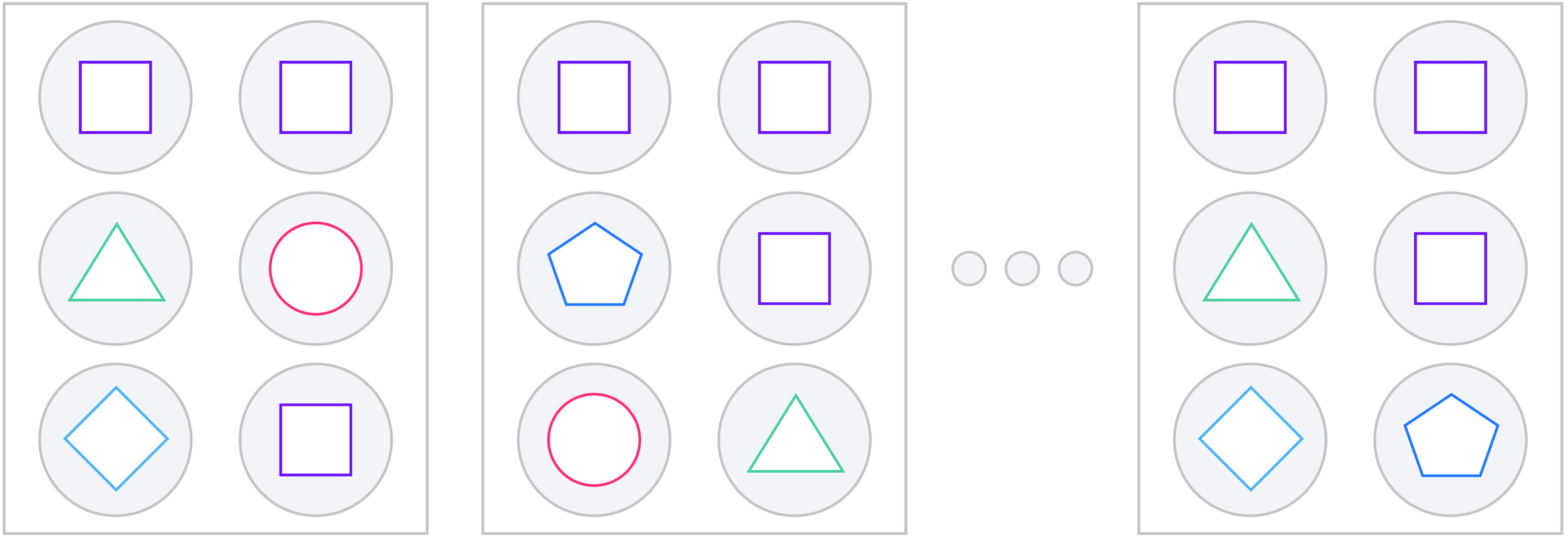


Техники работы с ошибками

Таймаут	не занимать лишние ресурсы
Прерыватель цепи	снизить нагрузку на сервисы, испытывающие проблемы
Повтор	получить ответ несмотря на сетевые проблемы
Идемпотентность	исключить бизнес-ошибки при повторах
Переборка	снизить влияние разного функционала друг на друга
Изоляция	снизить зависимость от других сервисов
Распределение нагрузки	обеспечить ресурсами критичные процессы
Балансировка	исключить единую точку отказа

Масштабируемость

Сервисы распределяются между серверами
в зависимости от потребностей



Простота развёртывания

- Небольшие изменения
- Частые выкладки
- Низкие риски
- Независимость

Простота замены

Закон Конвея

«Организации проектируют системы, которые копируют структуру коммуникаций в этой организации»

Итоги

- **Микросервисы** хорошо подходят для разработки крупных, сложных приложений

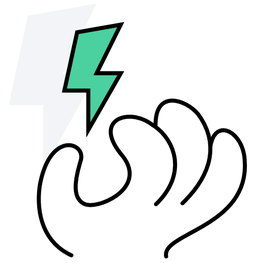
- Микросервисы накладывают повышенные требования к командам разработки и эксплуатации

Итоги

Микросервисы хорошо подходят для разработки крупных, сложных приложений

Микросервисы накладывают повышенные требования к командам разработки и эксплуатации

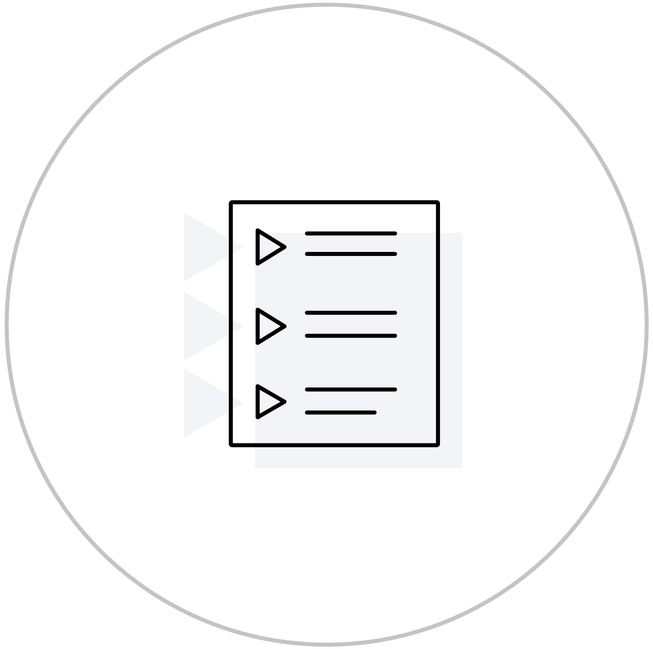
Сопутствующие проблемы



Михаил Триполитов
Банк Открытие, архитектор решений

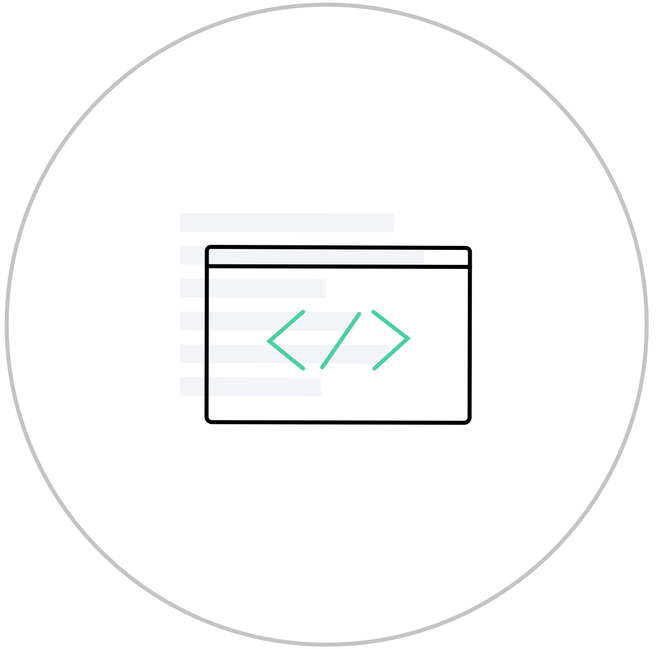
Тема занятия

1. Проблемы разработки
2. Проблемы эксплуатации



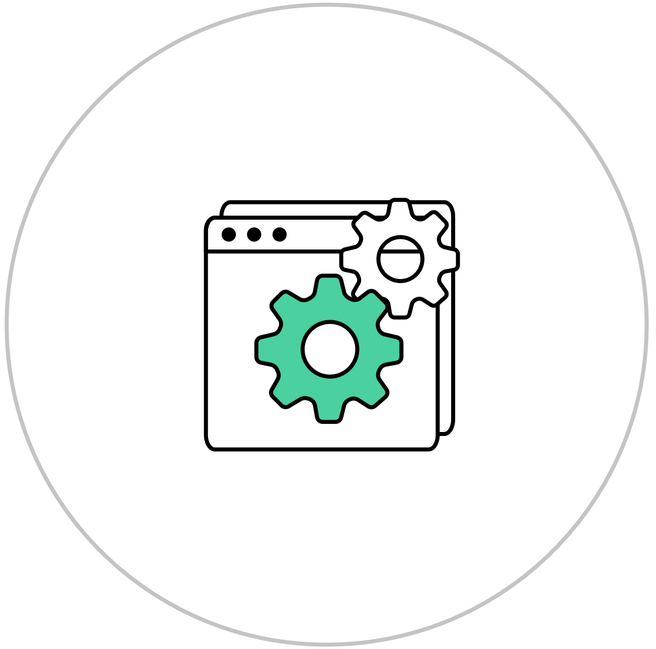
Проблемы разработки

- Совместимость API
- Версионирование артефактов
- Автоматизация сборки и тестирования
- Документация
- Инфраструктура разработки



Проблемы эксплуатации

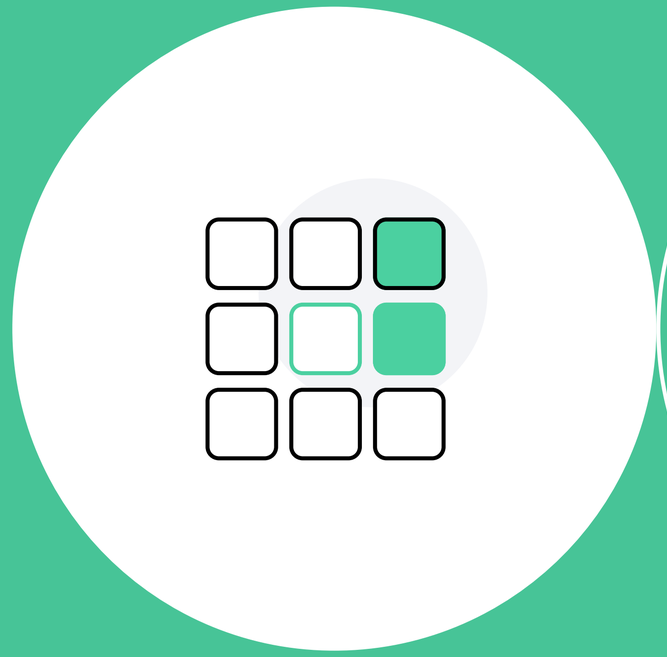
- Мониторинг
- Сбор логов
- Управление настройками
- Управление инфраструктурой



Итоги

Применение **микросервисной** архитектуры напрямую связано с проблемами применения и эксплуатации микросервисной архитектуры

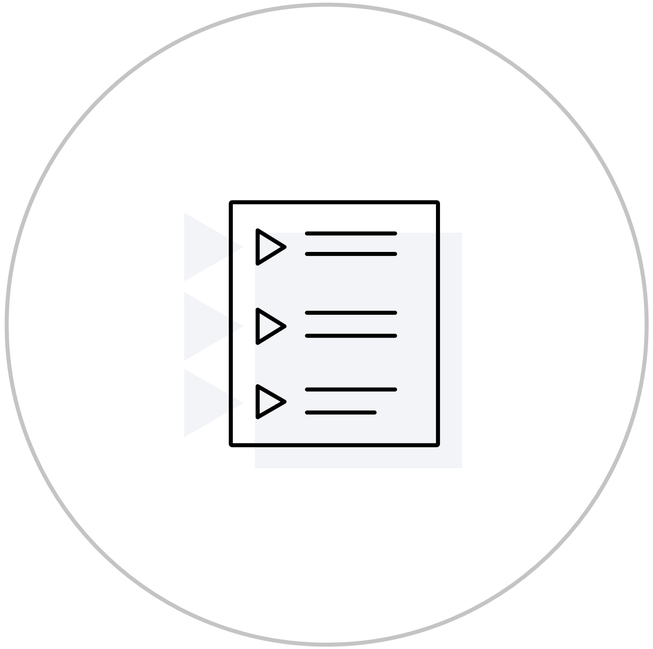
Антипаттерны



Михаил Триполитов
Банк Открытие, архитектор решений

Тема занятия

1. Антипаттерн: серебряная пуля
2. Антипаттерн: самоцель
3. Антипаттерн: наносервисы



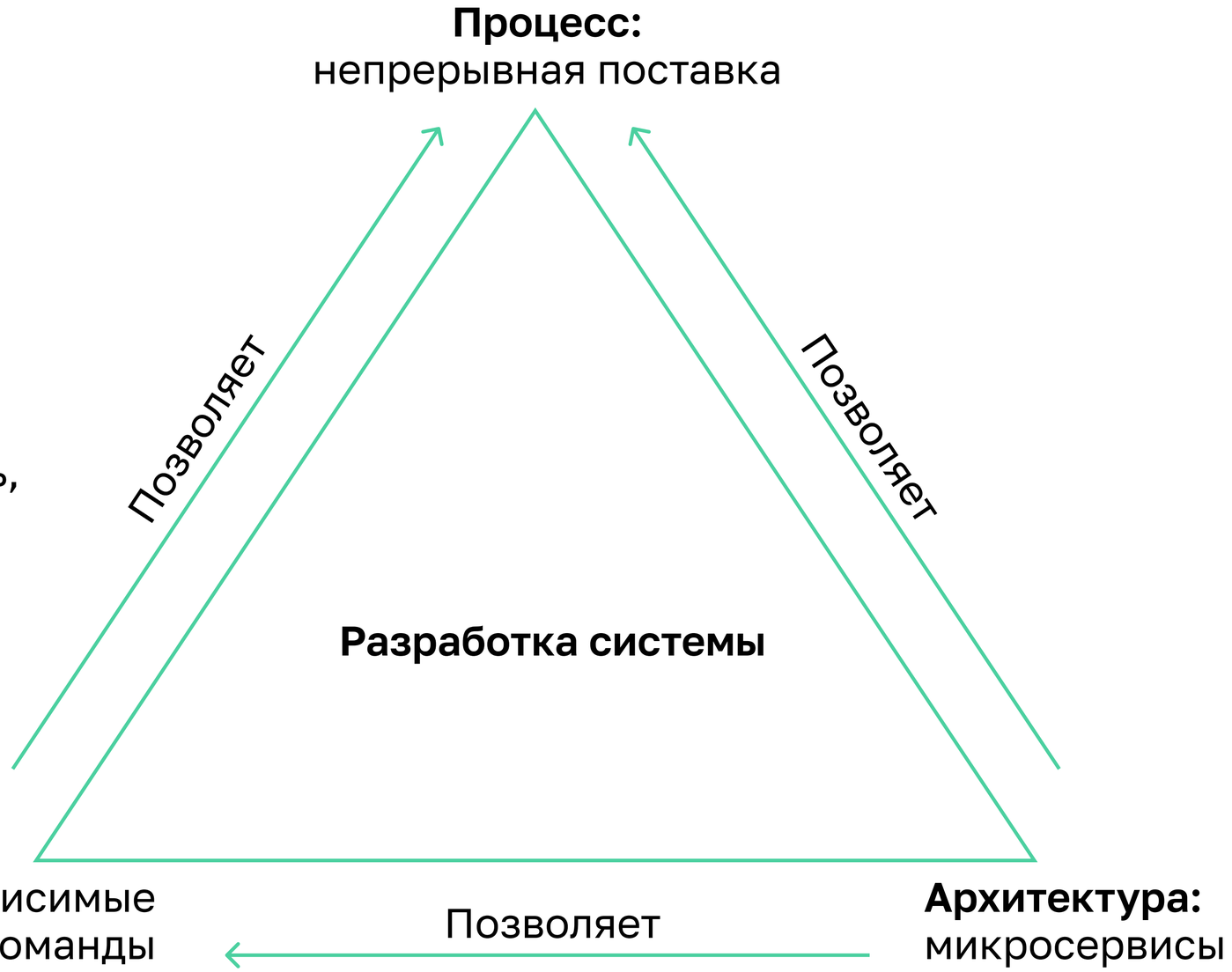
Антипаттерн: серебряная пуля

Нет

Пытаться решить
все проблемы разработки
применением микросервисов

Да

Микросервисы — архитектурный стиль,
который может помочь повысить скорость,
частоту и надежность релизов



Антипаттерн: самоцель

Нет

Делать внедрение микросервисов целью, по которой измеряется успех разработки IT-системы

Да

Цель — увеличить скорость, частоту и качество поставки

Хорошие метрики

- Время выкладки — время от коммита до выкладки
- Частота выкладки — количество выкладок в день на одного разработчика
- Интенсивность отказов — количество неуспешных выкладок
- Время восстановления — время восстановления после отказа

Антипаттерн: наносервисы

Нет

Создавать большое количество очень маленьких сервисов

Да

Один сервис на команду

Итоги

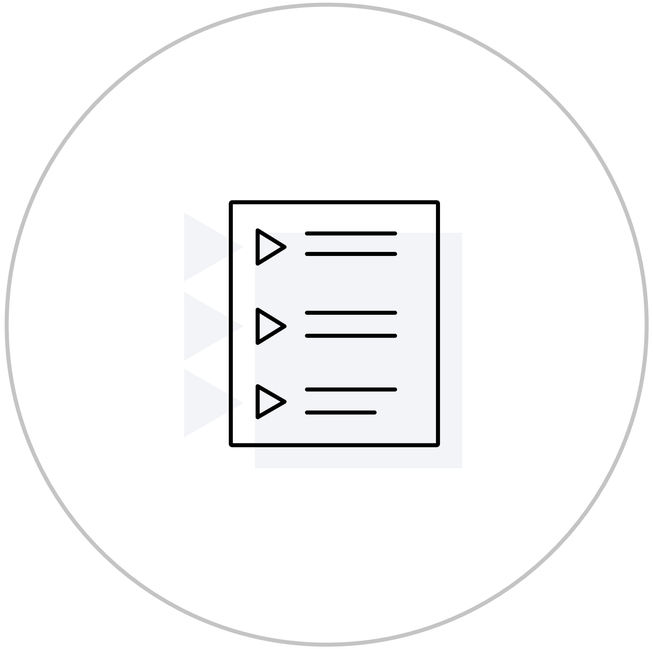
- Решение об использовании микросервисной архитектуры должно приниматься с чётким пониманием решаемых проблем и поставленной цели

Сложные решения

Михаил Триполитов
Банк Открытие, архитектор решений

Тема занятия

1. Разделение системы на сервисы
2. Кто владеет сервисом
3. Способ межсервисного взаимодействия
4. Когда и почему не стоит использовать микросервисы



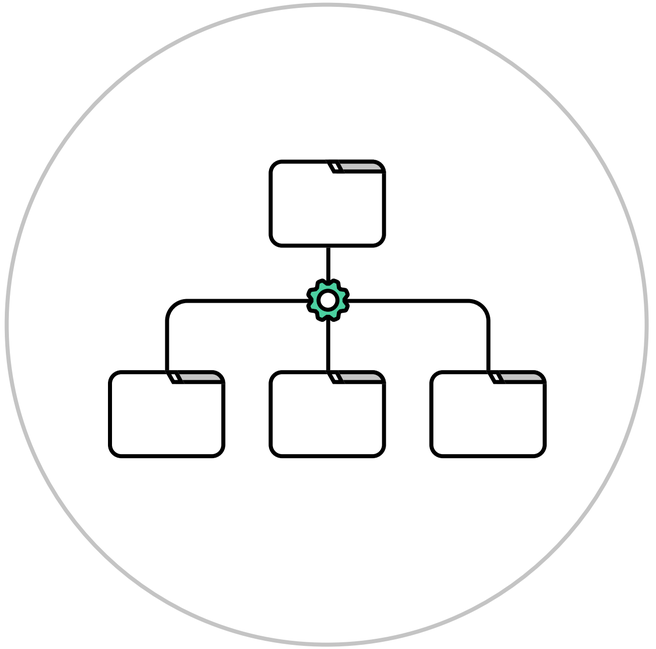
Разделение системы на сервисы

Low coupling and high cohesion:

- небольшое количество внешних связей
- решает близкие по смыслу задачи

Предметно-ориентированное проектирование:

ограниченный контекст → сервис



Владелец сервиса

1. Общий код → любой разработчик может изменить любой сервис и выложить его
2. По командам → только команда-владелец сервиса может изменить сервис и выложить
3. Общий код, но есть владелец сервиса → любой разработчик может изменить любой сервис, но выложить можно только по согласованию с владельцем сервиса

Синхронное и асинхронное взаимодействие

Синхронное
(Request/Response)

Да — простота понимания

Да — простота отладки и реализации

Нет — балансировка производительности

Нет — риск каскадных отказов

Нет — система балансировка нагрузки

Нет — организация Service Discovery

Асинхронное
(Event-based)

Да — устойчивость к пиковым нагрузкам

Да — балансировка нагрузки за счёт брокеров очередей

Да — слабая связность системы

Нет — общая высокая сложность системы

Нет — запросы на чтение требуют дополнительных посредников

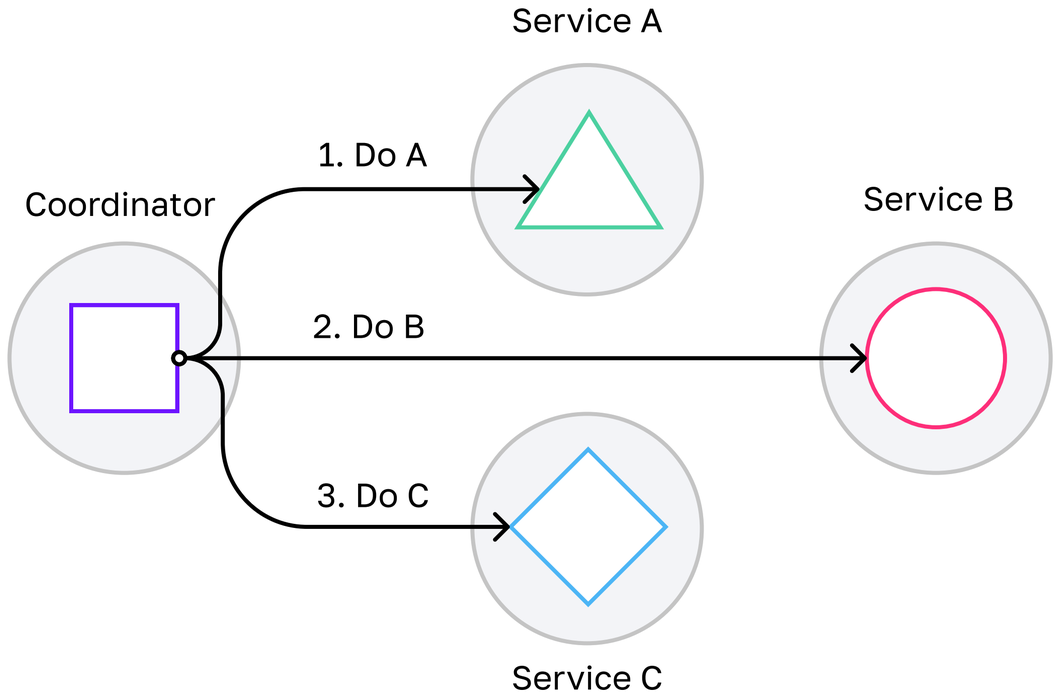
Оркестрация и хореография

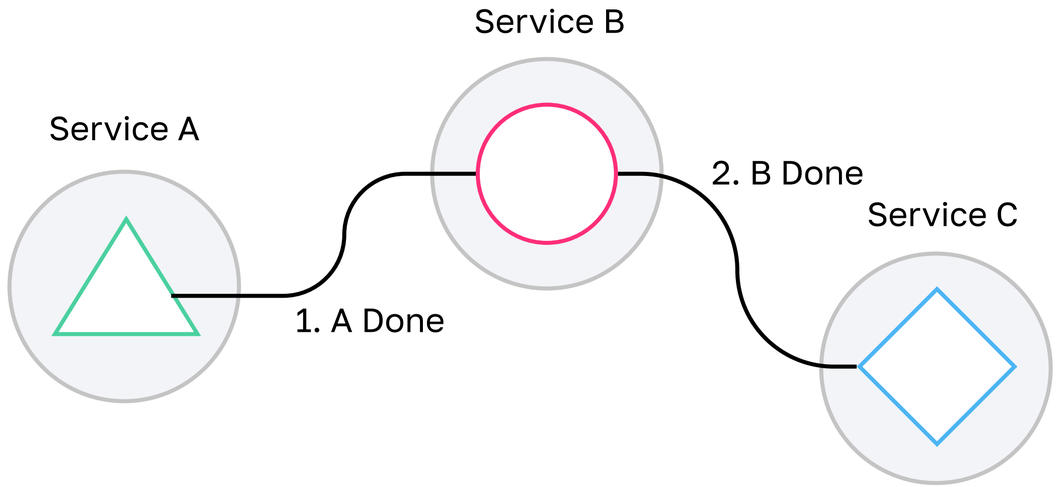
Оркестрация

Отдельный координатор управляет сервисами, указывая, какие операции выполнять в какой момент времени

Хореография

При выполнении операции каждый сервис публикует события, которые запускают операции в других сервисах





Протоколы интеграции

1. REST → производительный, масштабируемый, простой
2. Grpc → лёгкий, эффективный
3. GraphQL → адаптивный, эффективный, гибкий
4. JSON-RPC → лёгкий, понятный

Подход к безопасности

1	No authentication	доверять всем запросам внутри периметра
2	HTTP(S) Basic Authentication	передавать логин и пароль в заголовках запроса
3	Aouth2 и OpenID Connect	использовать SSO для межсервисного взаимодействия
4	Client Certificates and mutual TLS	каждому сервису выпускать свой клиентский сертификат
5	HMAC over HTTP	подписывать каждый HTTP-запрос секретным ключем
6	API Keys	каждому сервису выпускать API-ключ, по которому определять его права

Когда не стоит использовать

1. Незнакомая предметная область → чем меньше вы понимаете предметную область, тем сложнее найти границы контекстов

2. Не определена структура организации → разделение на сервисы не принесёт выгоды, если не будет соответствовать разделению на команды

3. Система с чистого листа → гораздо легче делить существующую систему на микросервисы, чем пытаться разделить на сервисы то, чего нет

Итоги

- При использовании микросервисов важно не только найти подход к разделению системы на сервисы, но и определиться с правилами владения сервисов
- Выбор способа и подходов межсервисного взаимодействия определит возможности всей системы
- Микросервисы стоит применять только тогда, когда выгоды от их использования превышают сложности применения подхода

Итоги

- При использовании микросервисов важно не только найти подход к разделению системы на сервисы, но и определиться с правилами владения сервисов
- Выбор способа и подходов межсервисного взаимодействия определит возможности всей системы
- Микросервисы стоит применять только тогда, когда выгоды от их использования превышают сложности применения подхода

Итоги

- При использовании микросервисов важно не только найти подход к разделению системы на сервисы, но и определиться с правилами владения сервисов
- Выбор способа и подходов межсервисного взаимодействия определит возможности всей системы
- Микросервисы стоит применять только тогда, когда выгоды от их использования превышают сложности применения подхода