



Микросервисная архитектура, ПОДХОДЫ И ТЕХНОЛОГИИ

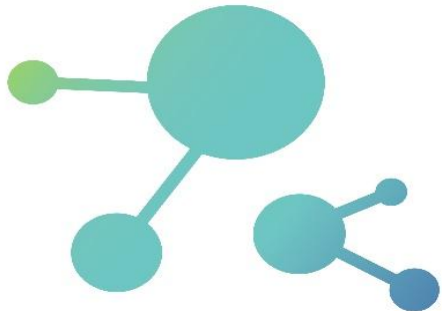
Ветчинкин Кирилл

<https://www.facebook.com/k.vetchinkin>

k.vetchinkin@yandex.ru

О себе

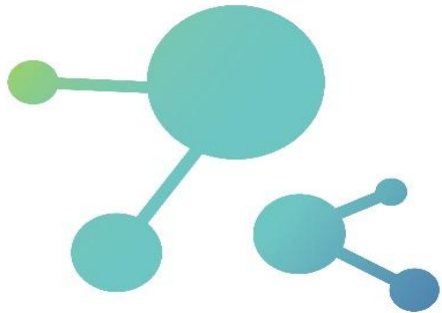
**PAY
SYSTEM
.TECH**



- Руководитель разработки, архитектор
- Практикую:
 - IT strategy
 - Agile
 - DevOps
 - Микросервисы

О компании

**PAY
SYSTEM
.TECH**



- Финтех-проекты
- Платежные системы
- Крупные интеграции

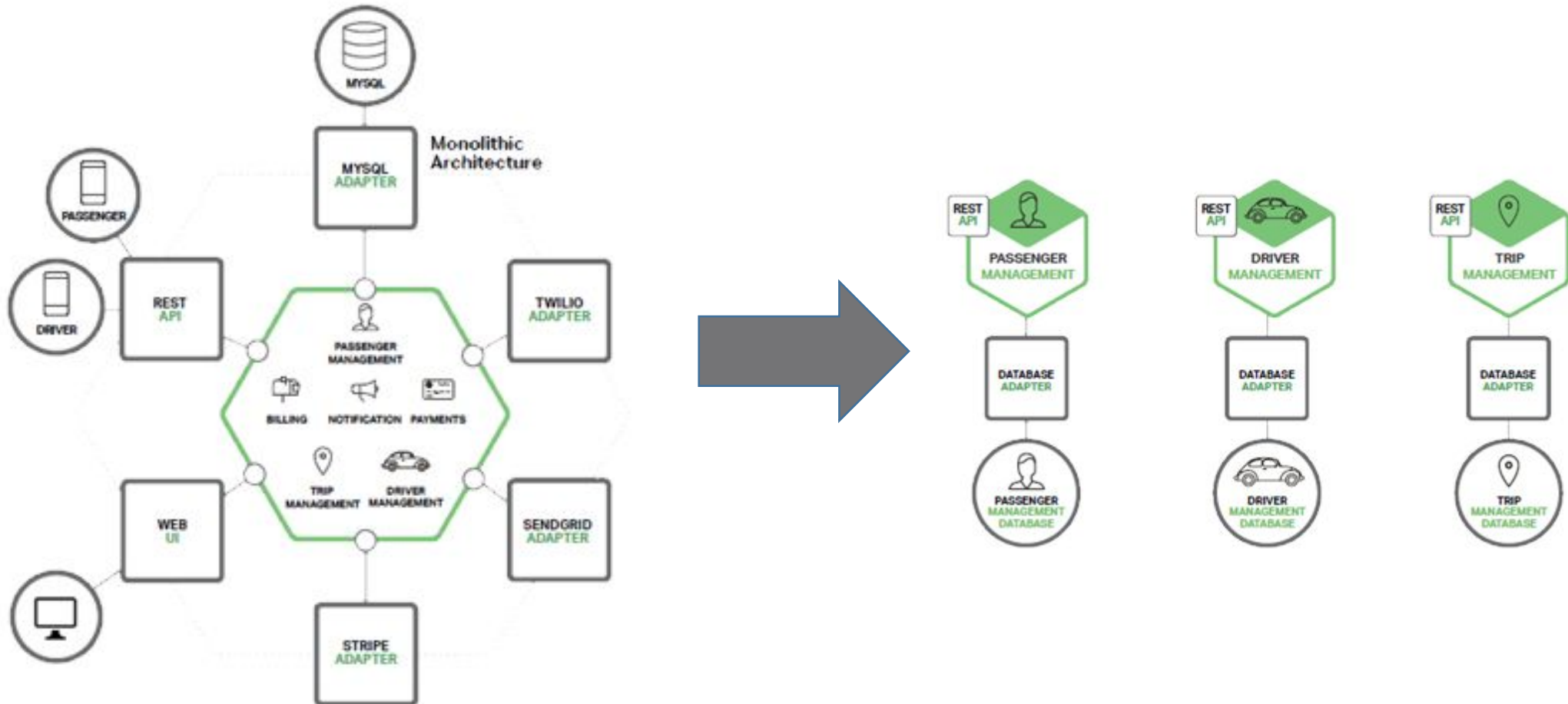
О чем доклад?

- Вспомним кратко идею подхода
- Нужны ли вам микросервисы?
- Архитектура системы
- Паттерны/антипаттерны
- Проблемы и решения
- Развертывание и технологии

Проблематика больших монолитных систем

- Плохое горизонтальное масштабирование
- Плохая отказоустойчивость
- Сложность внедрения новых технологий
- Сложность рефакторинга legacy
- И т.п....

Суть микросервисного подхода



Плюсы и минусы

Плюсы

- Горизонтальное масштабирование
- Отказоустойчивость
- Масштабирование команд
- Переиспользование
- Гибкость стека

Минусы

- Сложно
- *Дорого*
- Не согласованные данные

Вам не нужны микросервисы, если

- Вы делаете стартап (MVP)
- Нет или не предполагается рост нагрузки

Вам нужны микросервисы, если

- Высокая нагрузка
- Система растет
- Команда растет
- Нужна отказоустойчивость
- Необходимо сократить TTM

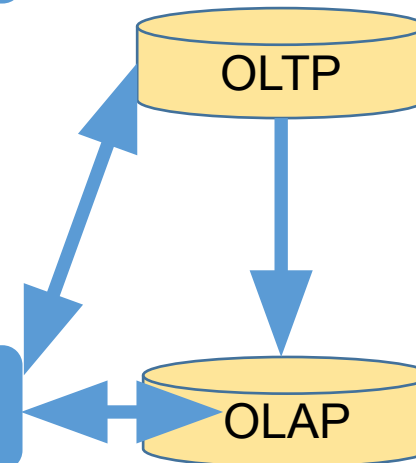
Разрез на сервисы

Контракт

Логика

- Платежи
- История платежей
- Акции
- Sms
- Отчеты

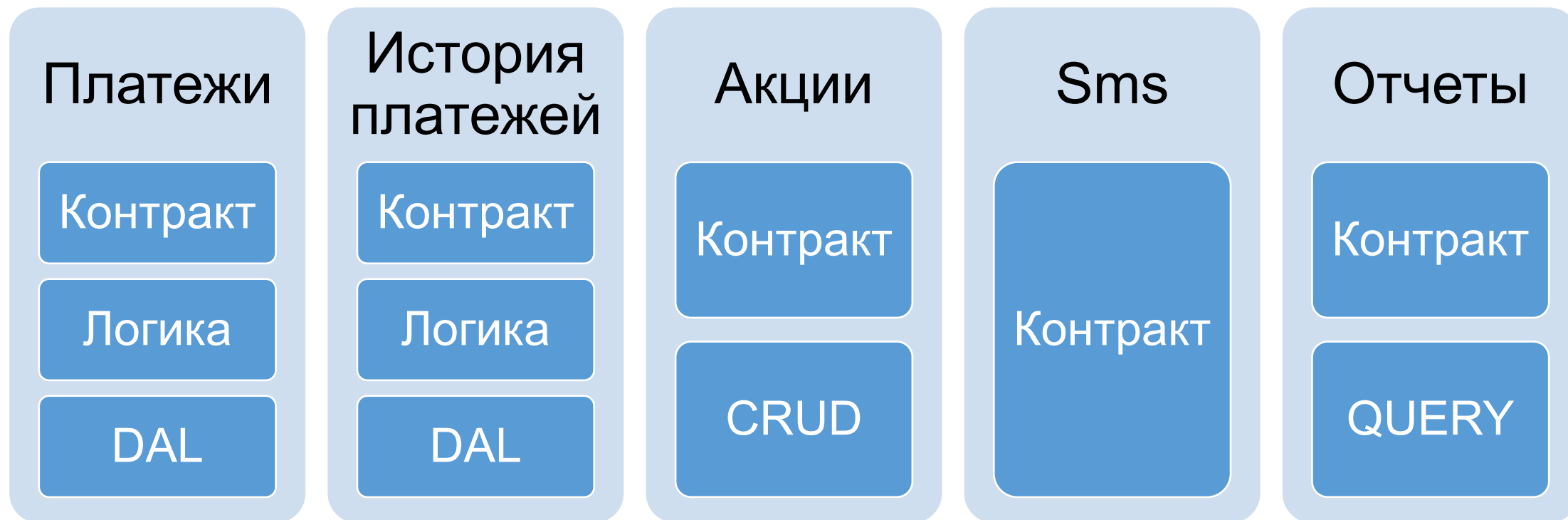
Dal



Разрез на сервисы

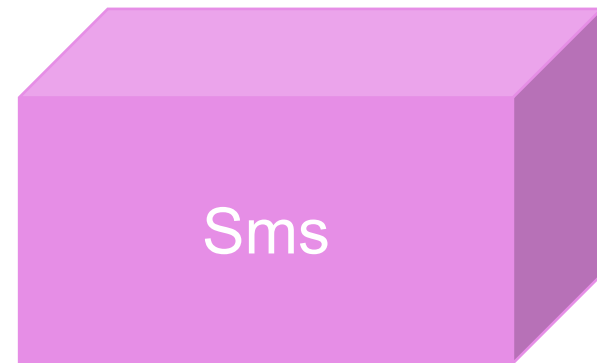
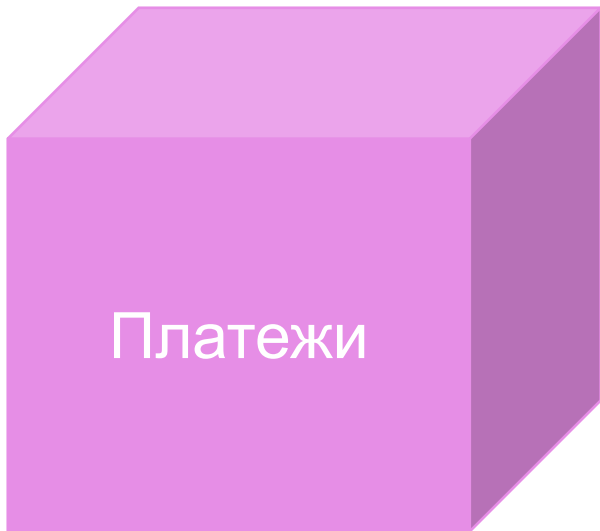


Не разрезайте по слоям, разрезайте по бизнес-контекстам



Размер сервиса

- На основе бизнес-контекста
- На основе сетевых запросов
- На основе транзакций

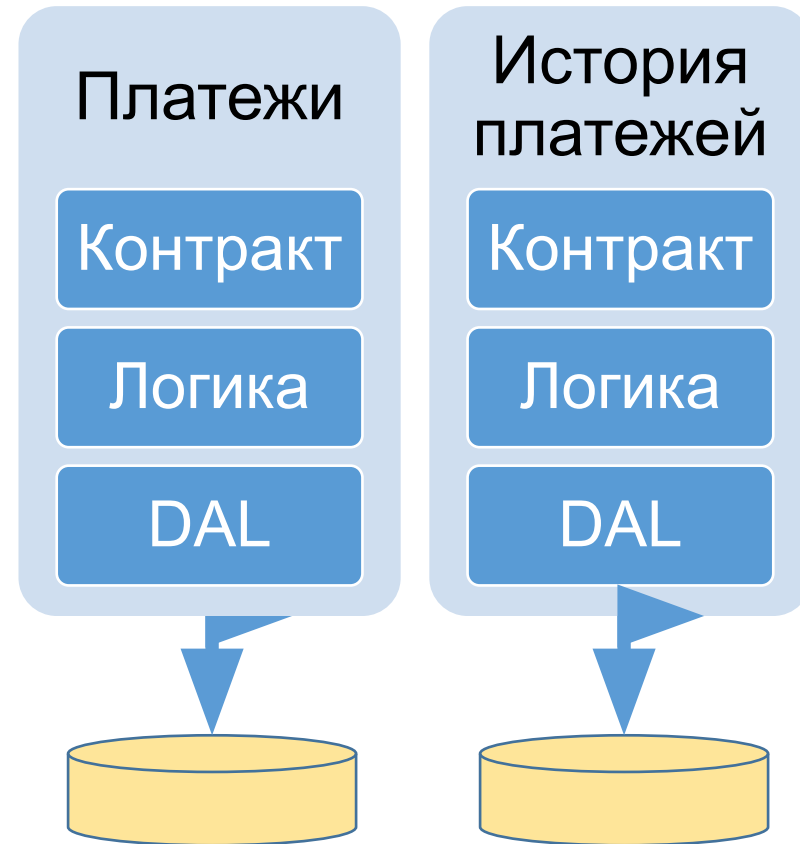


Один VCS/CI/CD на сервис

- Один сервис – один репозиторий
- Один сервис – один CI-конвейер
- Один сервис – один CD-конвейер

База данных

- Один сервис – одна база
- Выбор типа базы данных зависит от задачи



Events или RPC



Events(AMQP, MSMQ и т.п)

- Асинхронное взаимодействие

RPC(HTTP REST)

- Синхронное взаимодействие

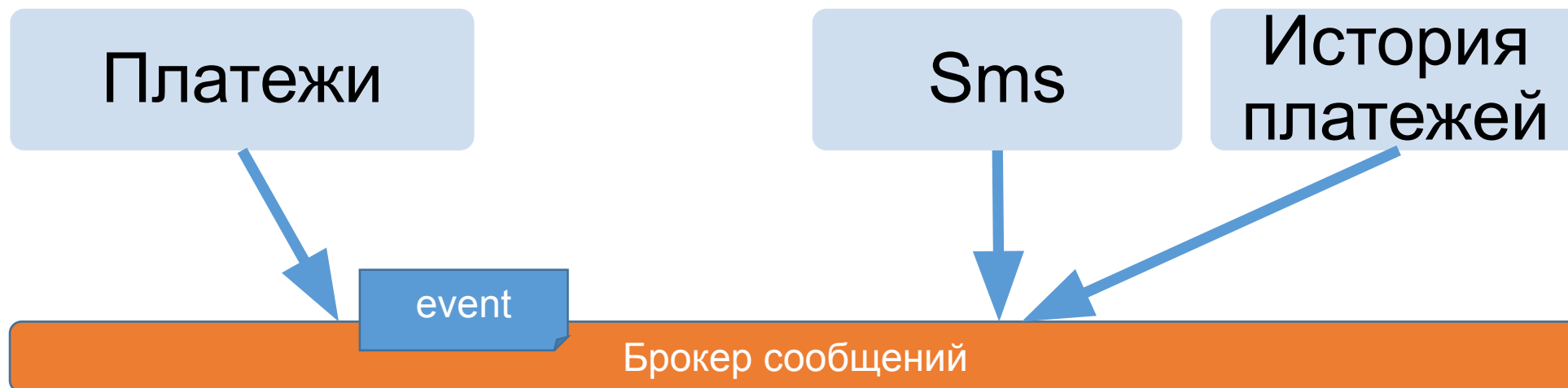
Выбор протокола обмена сообщениями

- Специфические для платформы (JMS, MSMQ)

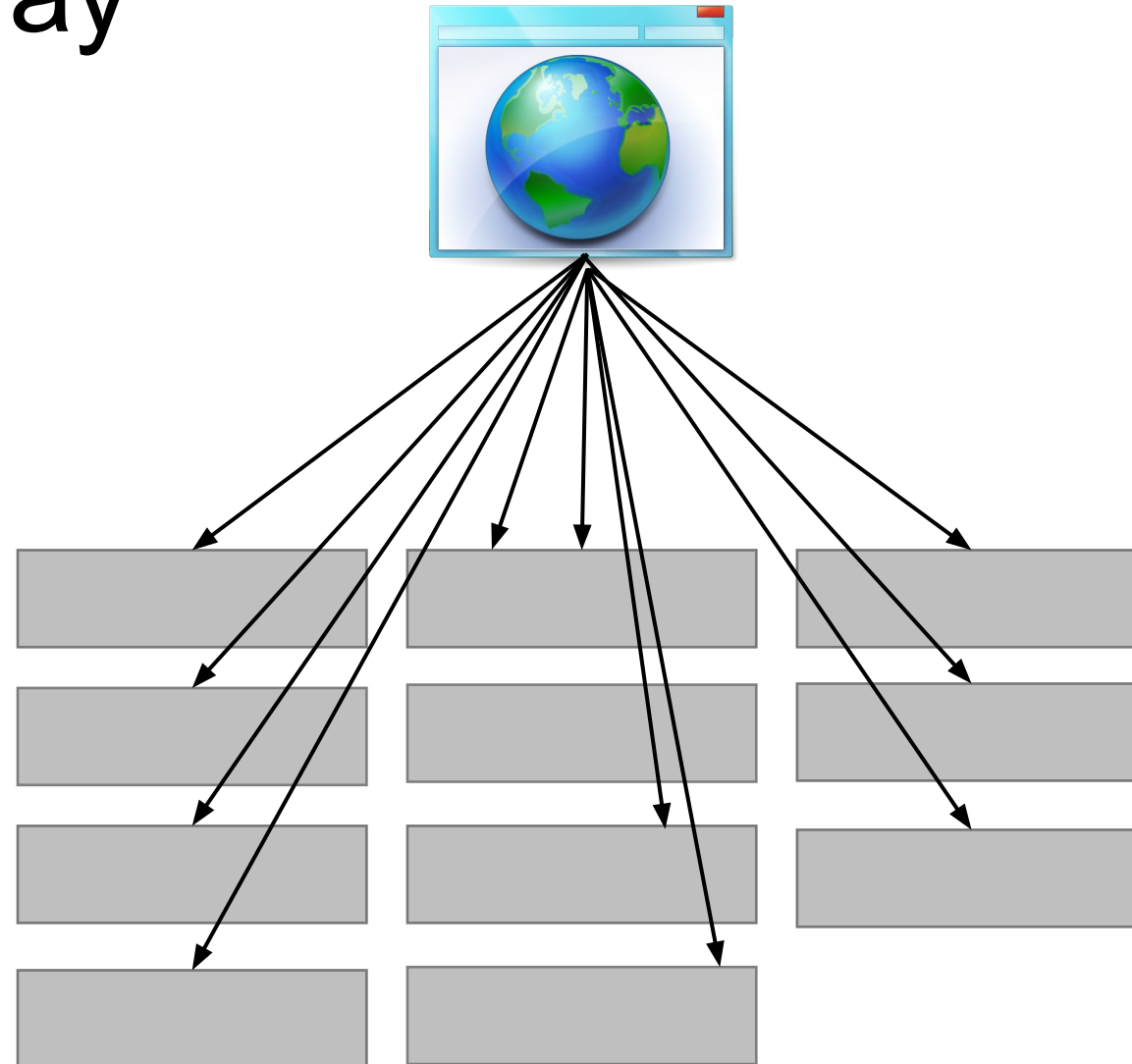


- Независимые от платформы стандарты (AMQP и прочие)

Брокер сообщений



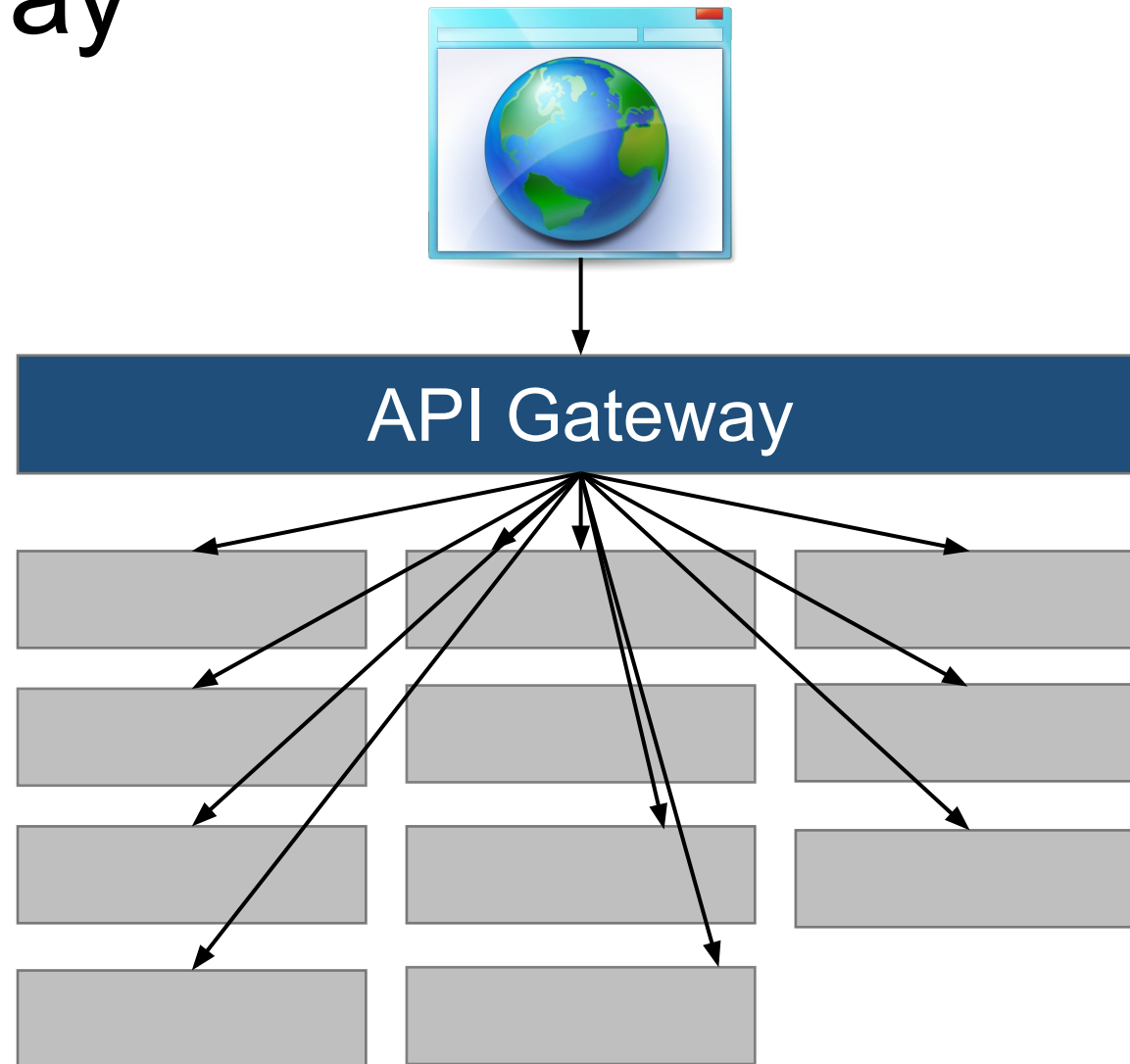
ApiGateway



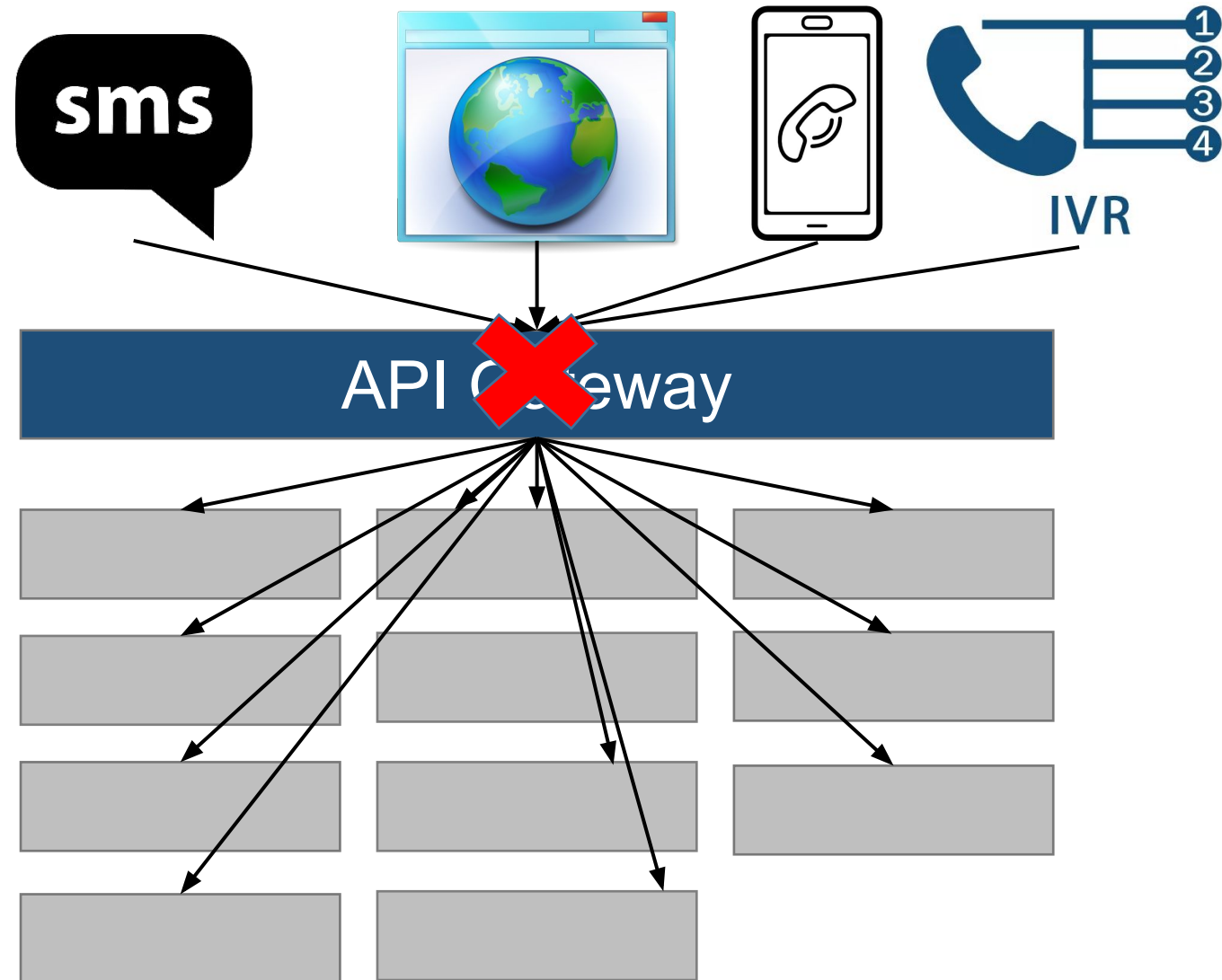
ApiGateway

- Единая точка входа для клиента
- Нет логики
- Делается под клиента

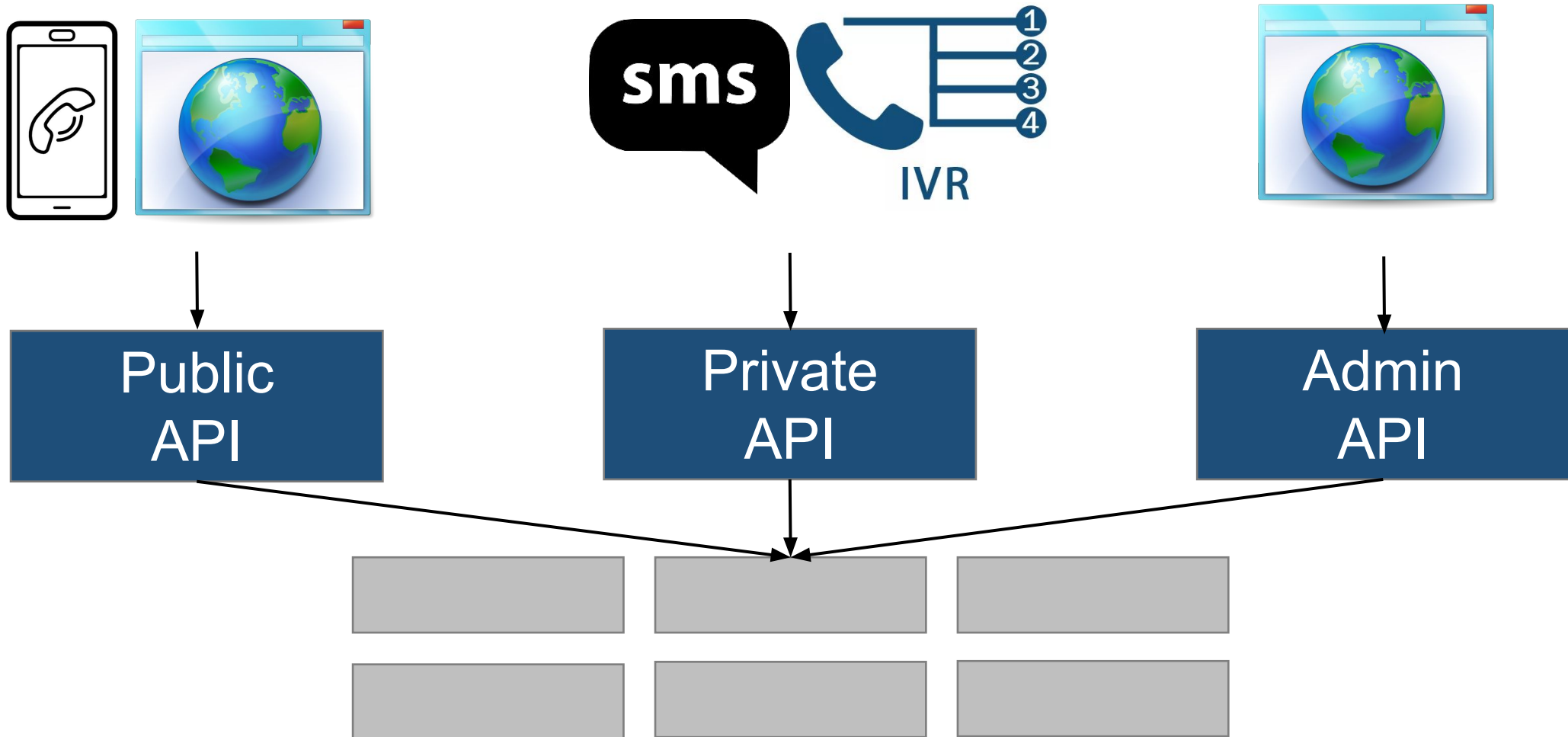
ApiGateway



BFF



BFF

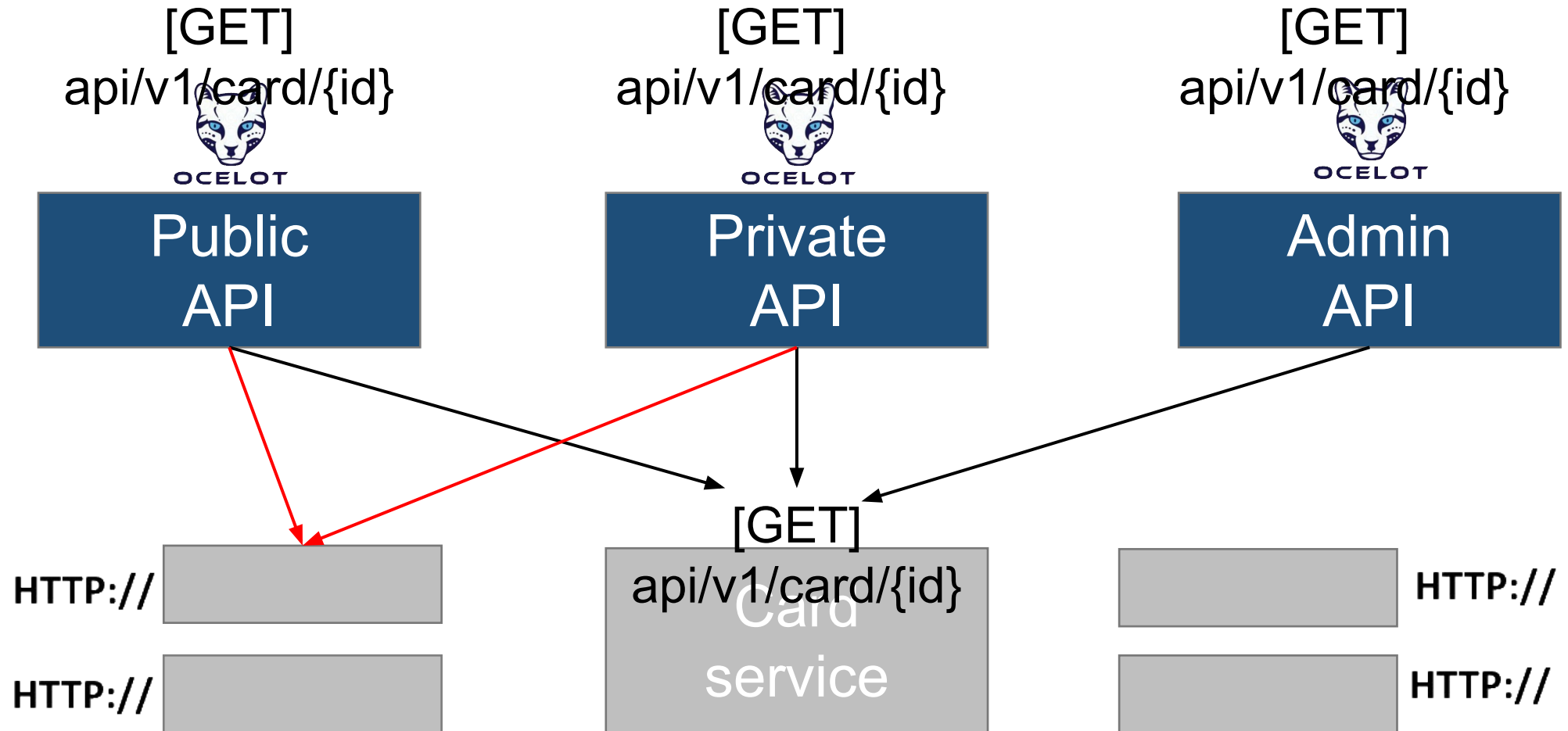


HTTP Proxy(Ocelot)

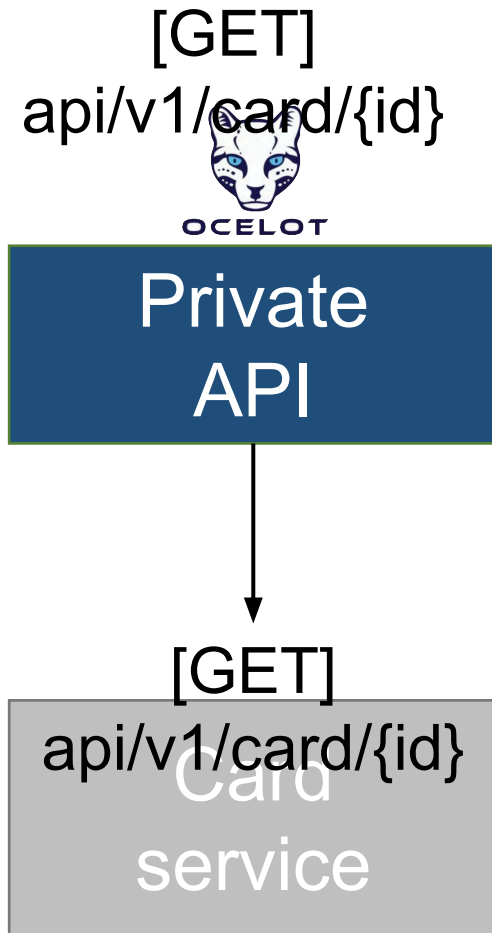


<https://github.com/ThreeMammals/Ocelot>

HTTP Proxy(Ocelot)

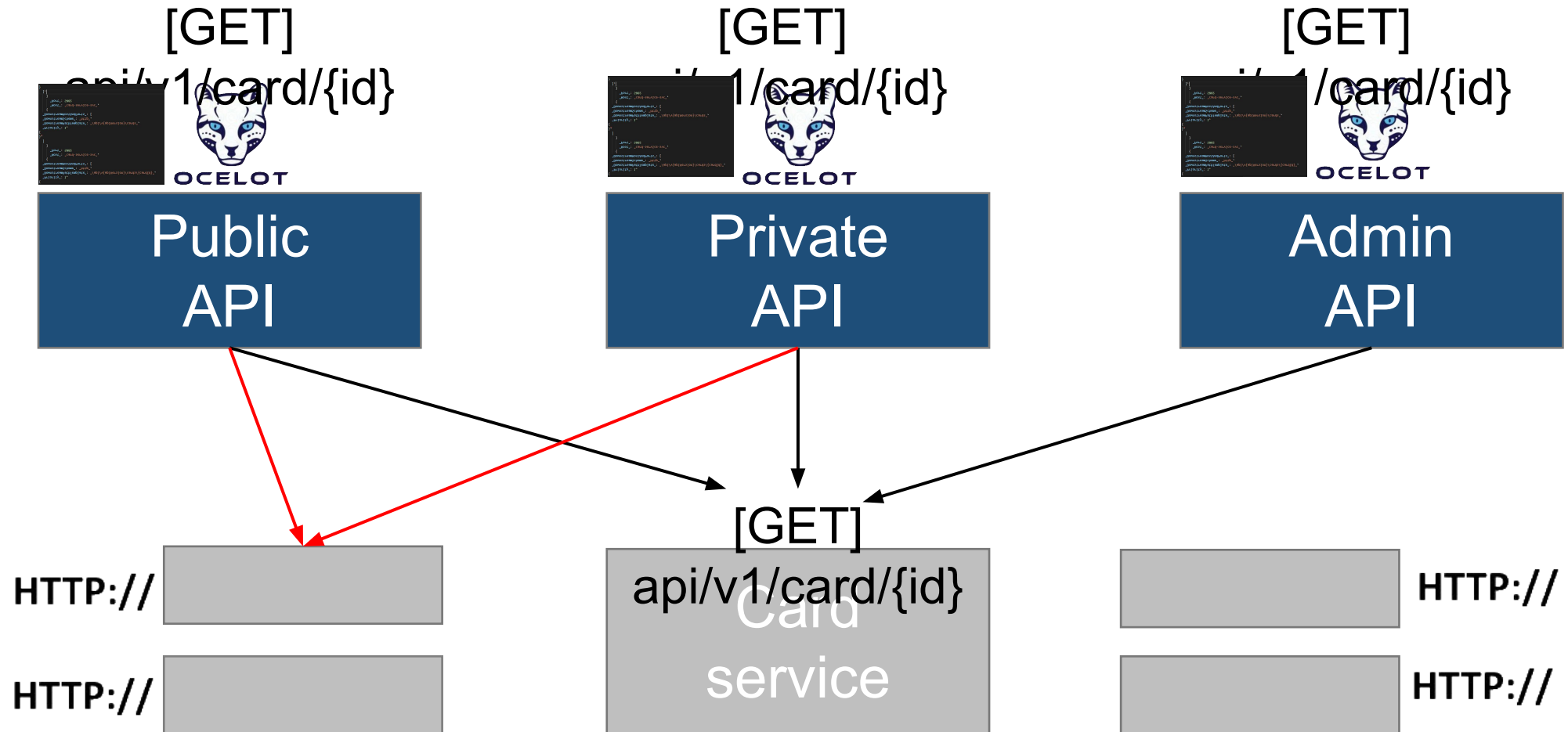


HTTP Proxy(Ocelot)

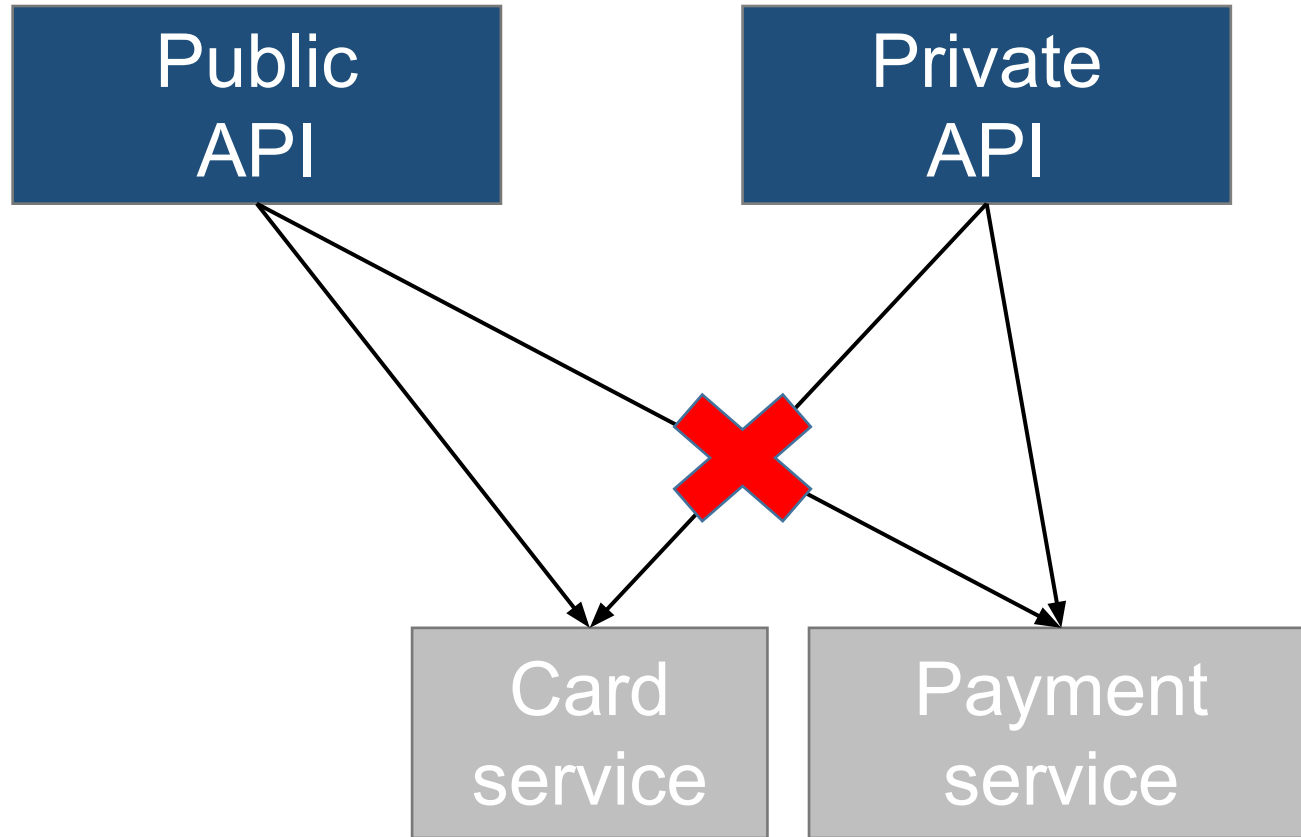


```
{
  "Priority": 1,
  "DownstreamPathTemplate": "/api/v{apiVersion}/cards/{cardId}",
  "DownstreamScheme": "http",
  "DownstreamHostAndPorts": [
    {
      "Host": "card-service-svc",
      "Port": 5002
    }
  ]
},
{
  "Priority": 1,
  "DownstreamPathTemplate": "/api/v{apiVersion}/cards",
  "DownstreamScheme": "http",
  "DownstreamHostAndPorts": [
    {
      "Host": "card-service-svc",
      "Port": 5002
    }
  ]
},
]
```

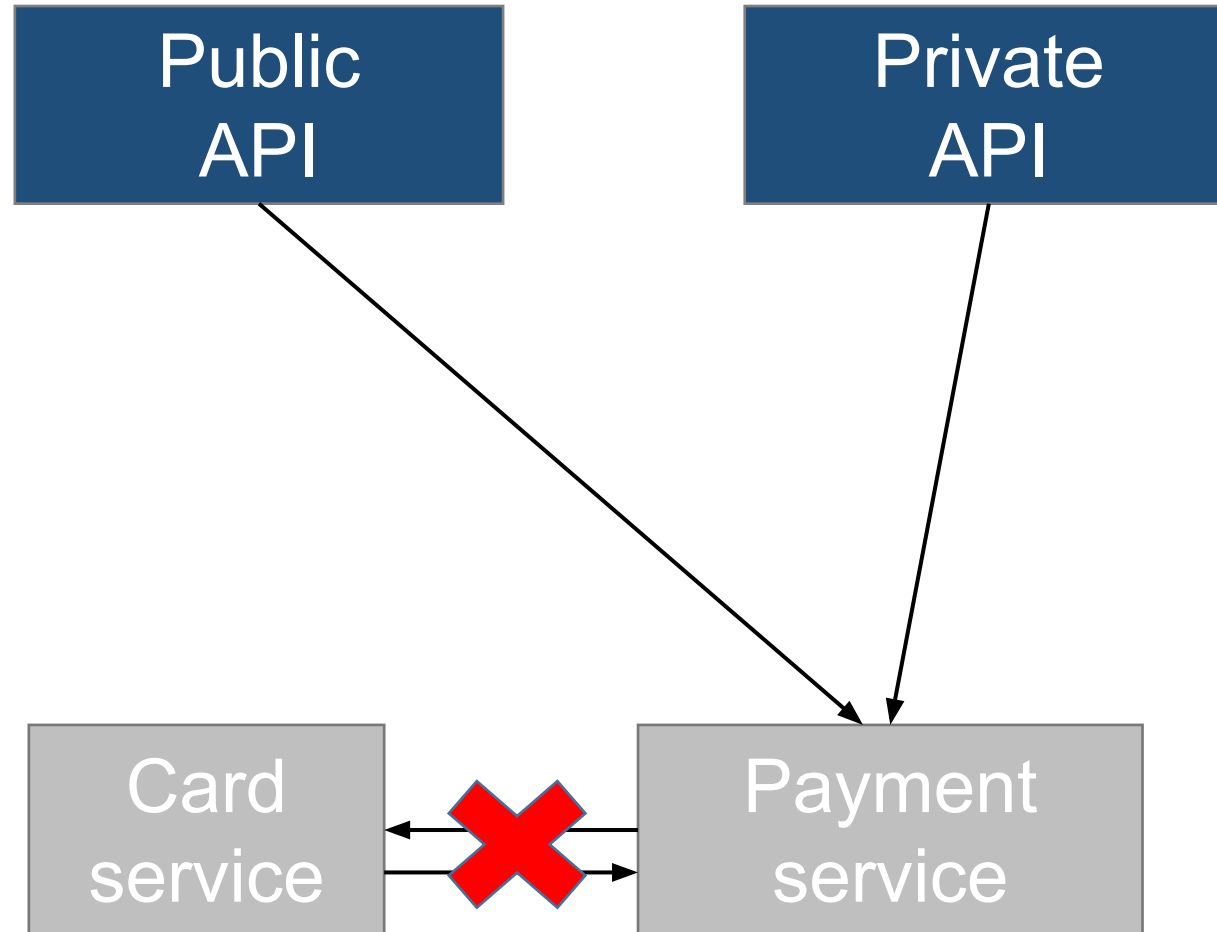
HTTP Proxy(Ocelot)



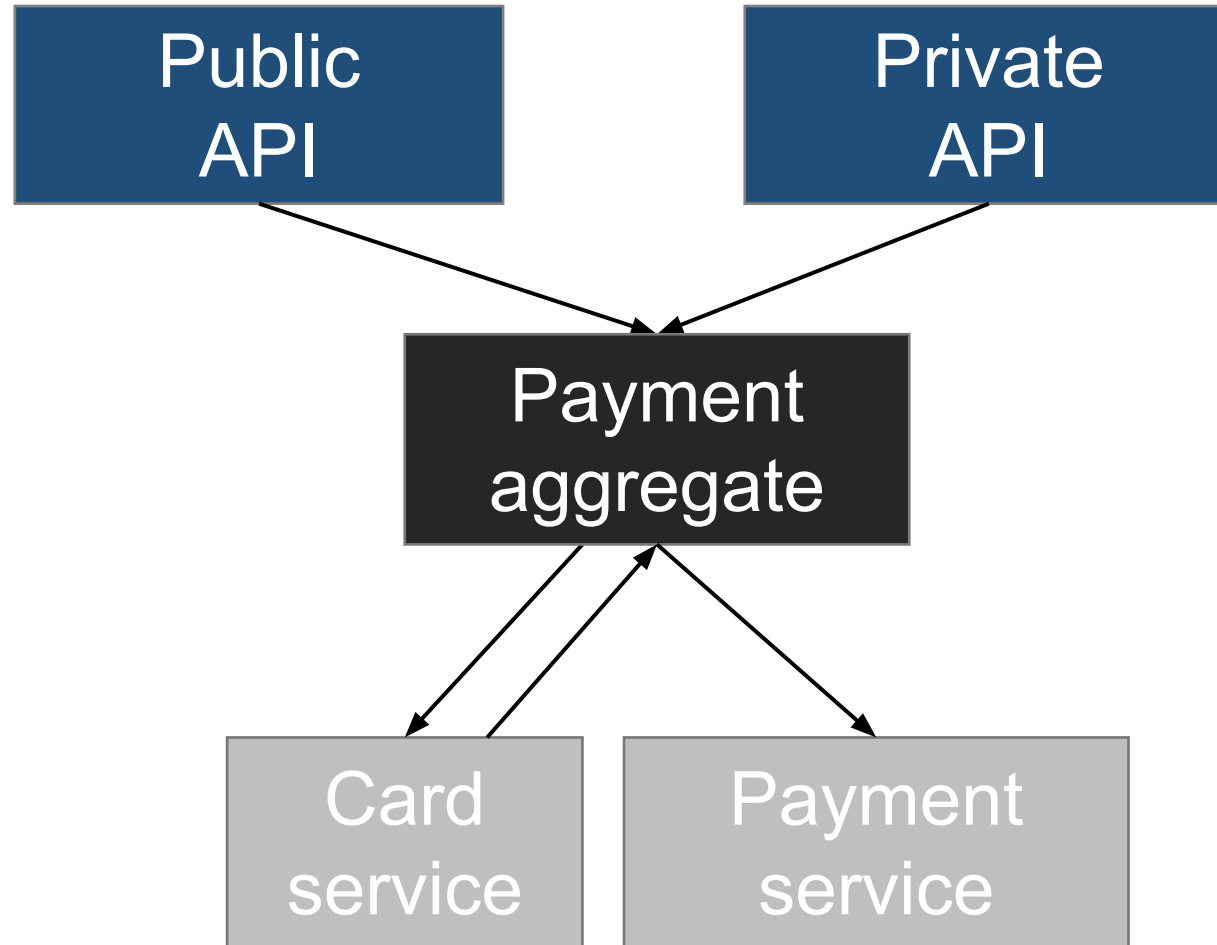
Платеж с дефолтной карты



Сильная связанность



Aggregate



Архитектура сервиса

Умный - глупый

Payment
service

CQS

DDD

Repository

Domain

events

Sms
service

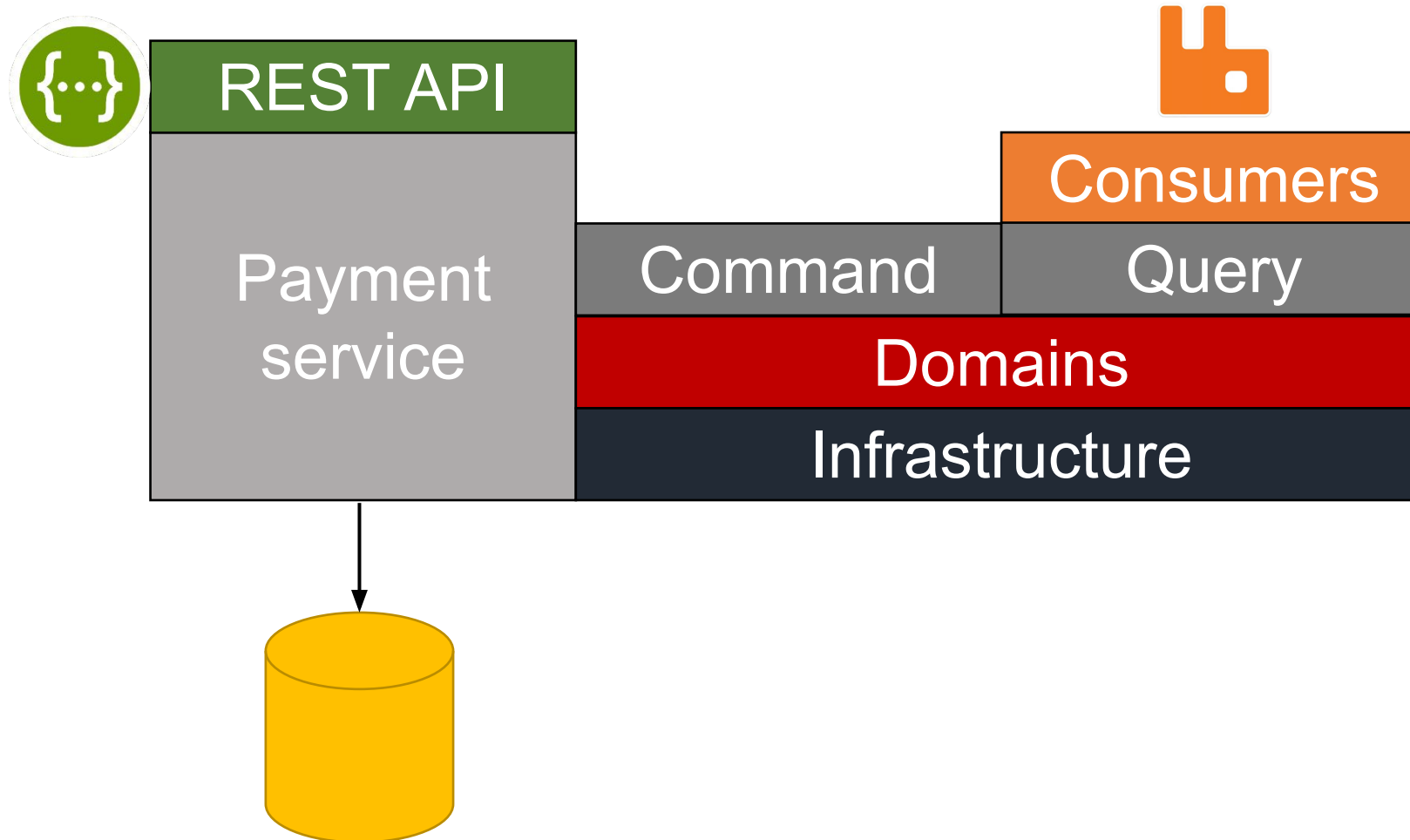
30 строк

кода

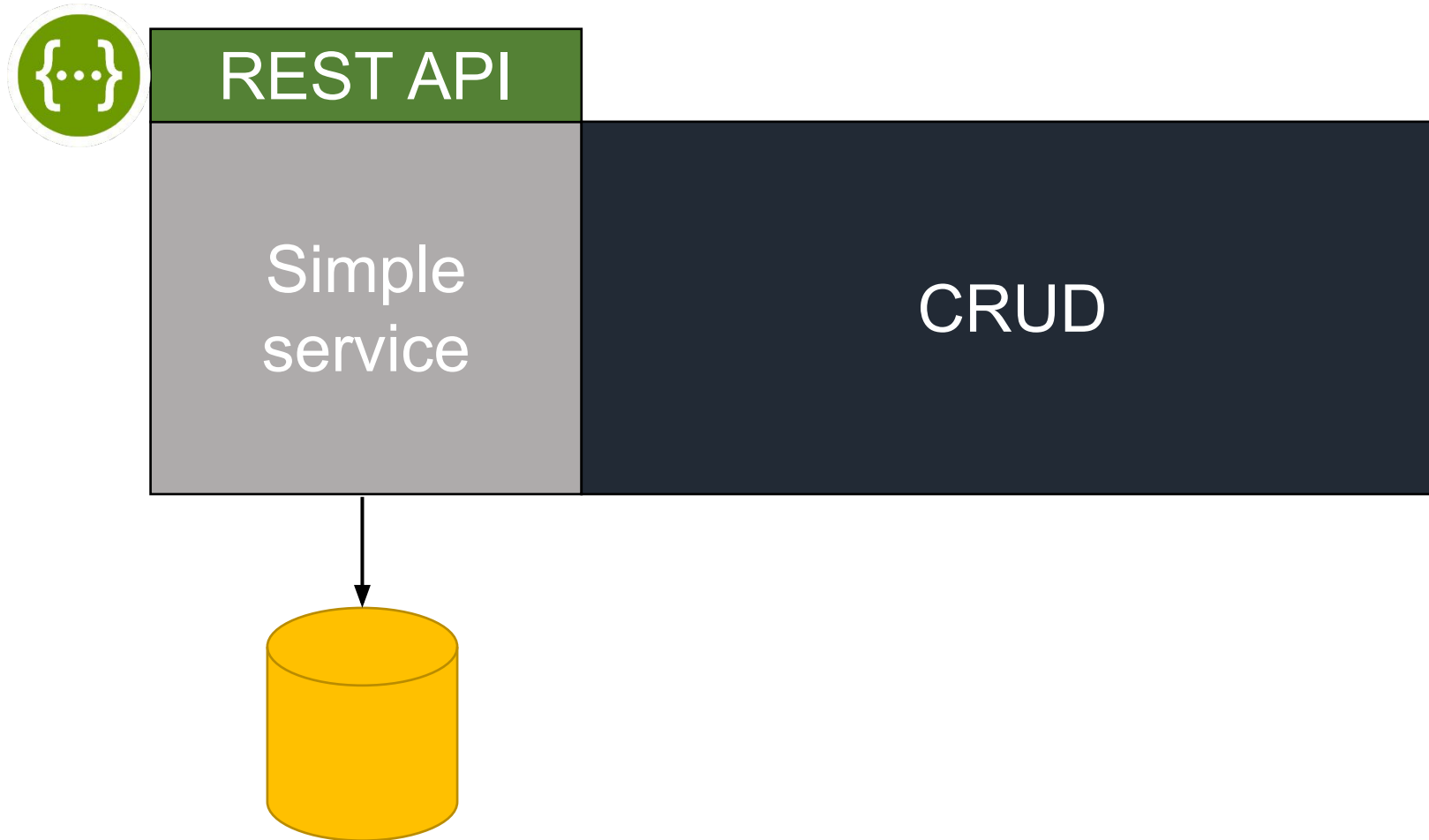
прямо в

Consumer

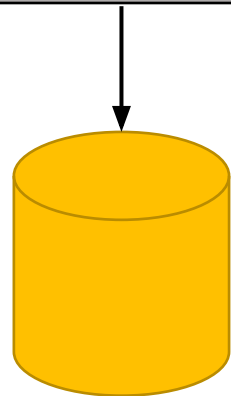
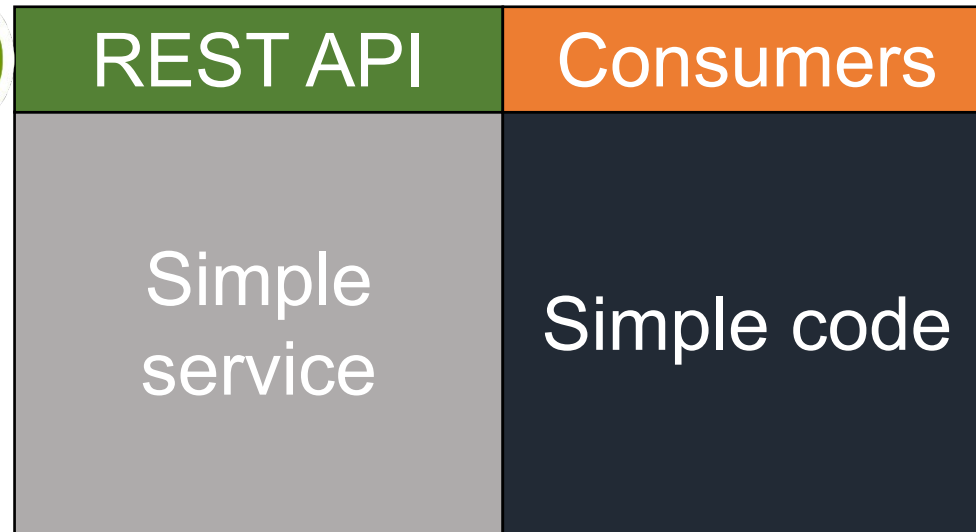
Структура умного сервиса



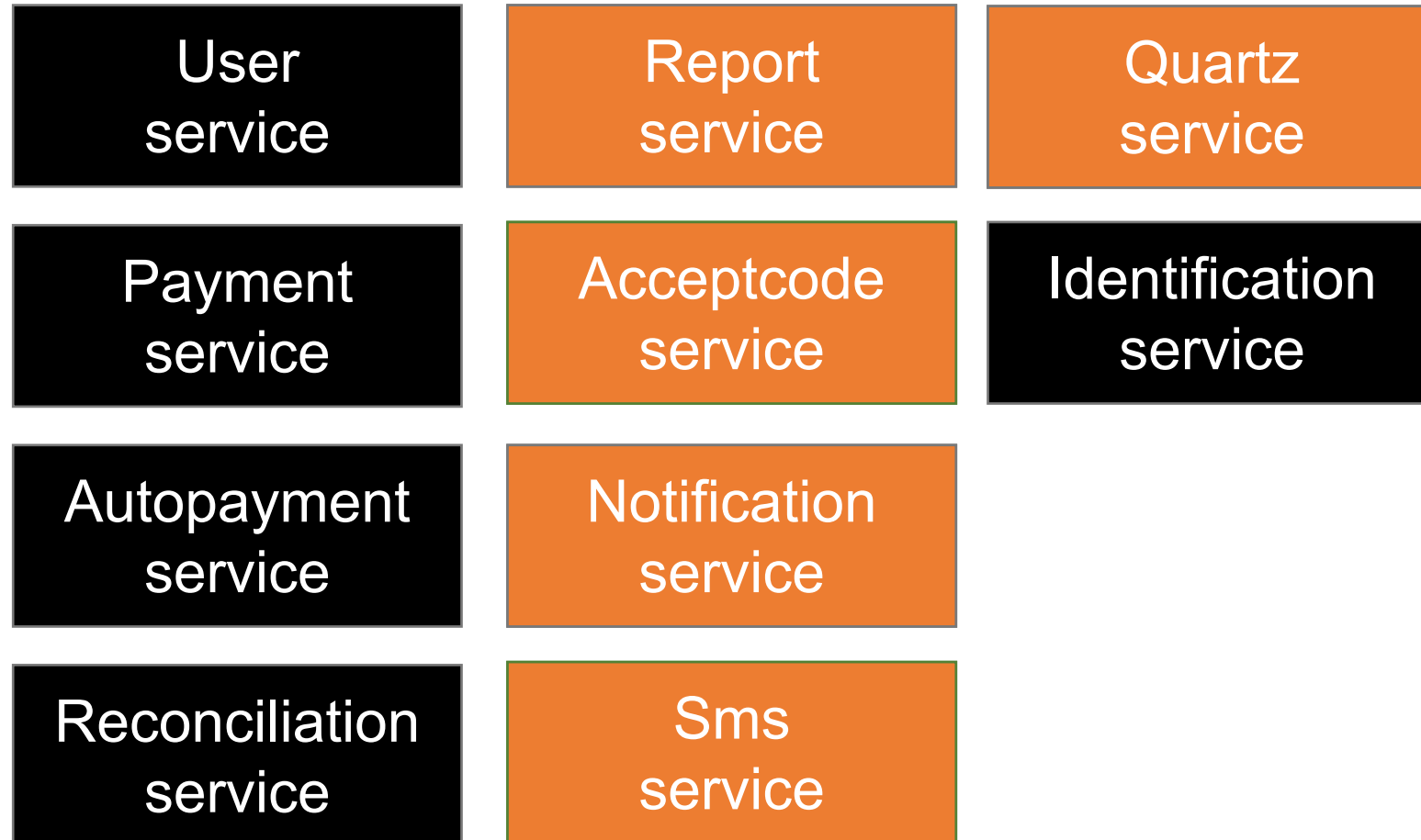
Структура простого сервиса



Структура простого сервиса



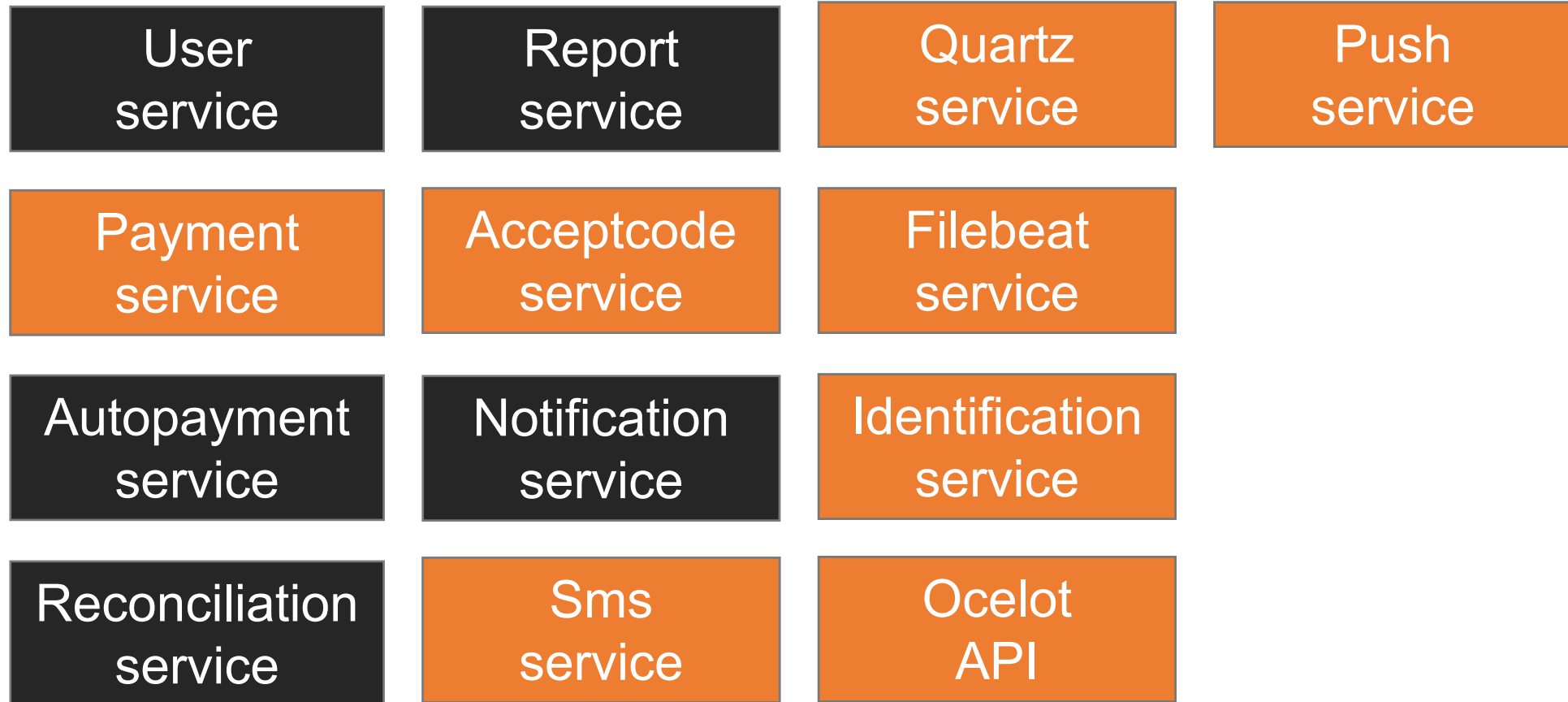
Умные и простые



Переиспользуемые

- Сервис решает типовые задачи
- В идеале это контейнер

Пример



Конфигурация через окружение

Система 1

env: ip = smsc.ru

Sms

v1.0.37

Система 2

env: ip = tele2.ru

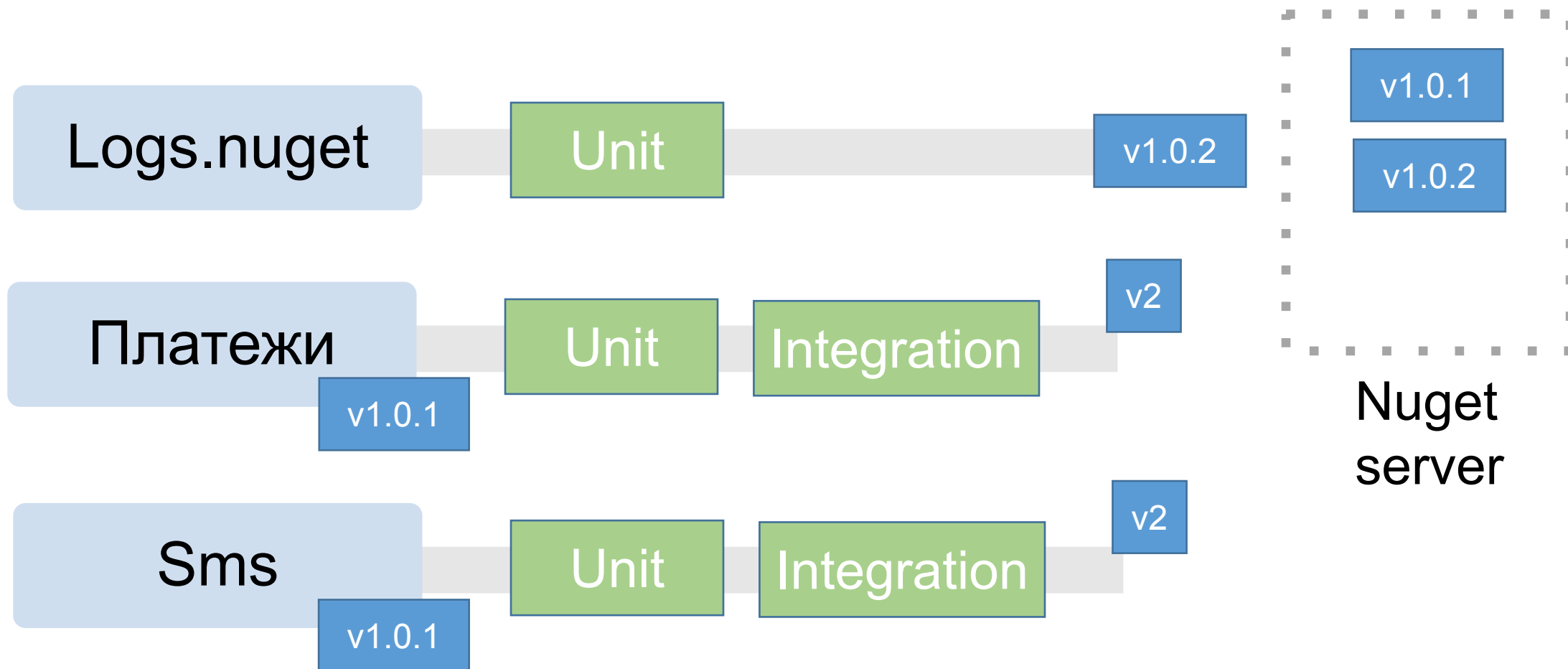
Sms

v1.0.37







Общий код

- Нарушение DRY – нормально
- Nuget для того, что фундаментально и не меняется

Общий код

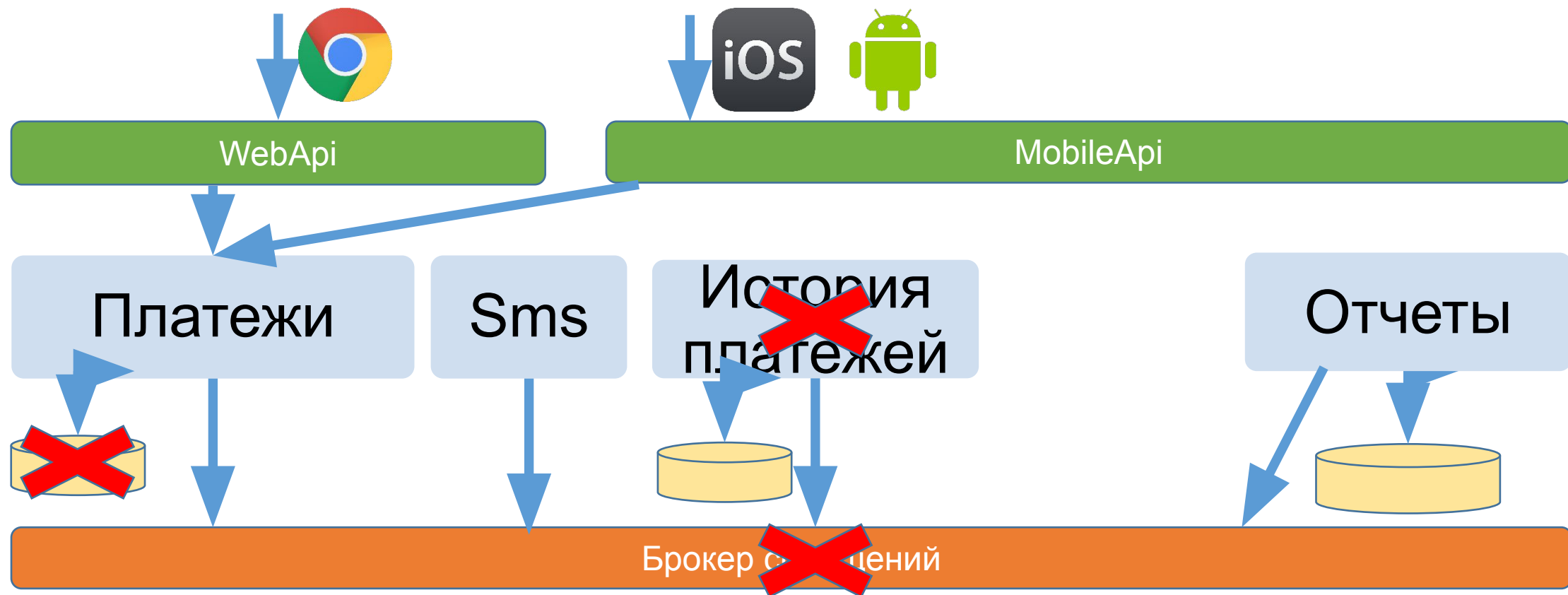


Общий код

- 🔖  Tume / NuGet / Tume.MassTransit.TransactionConsistency
Позволяет создавать согласованную интеграцию сообщений для микросервисов
- 🔖  Tume / NuGet / Tume.Ddd.Primitives
Базовые интерфейсы и абстрактные классы для Domain Driven Design архитектуры
- 🔖  Tume / NuGet / Tume.Cryptography
Пакет для шифрования и дешифрования
- 🔖  Tume / NuGet / Tume.OpenIso8583
Реализация протокола Iso8583
- 🔖  Tume / NuGet / Ftp.Transfer
Пакет работы FTP(Загрузка)
- 🔖  Tume / NuGet / Tume.MassTransit.QuartzJobService
Пакет для создания Job в планировщике Quartz

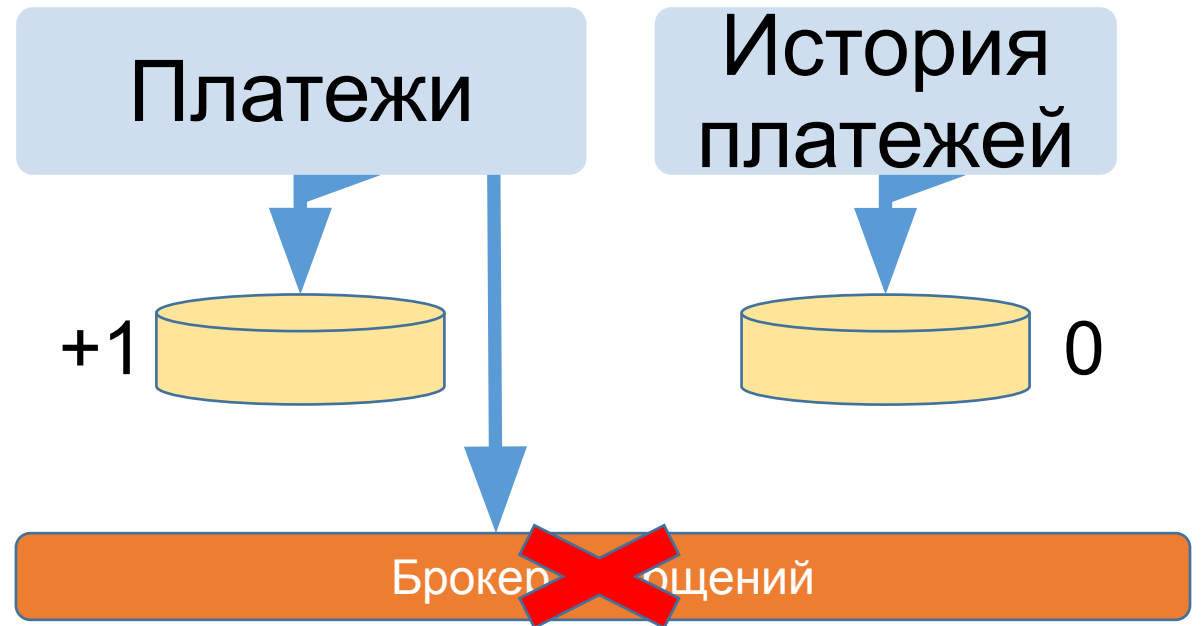
Проблемы и решения

Проблемы

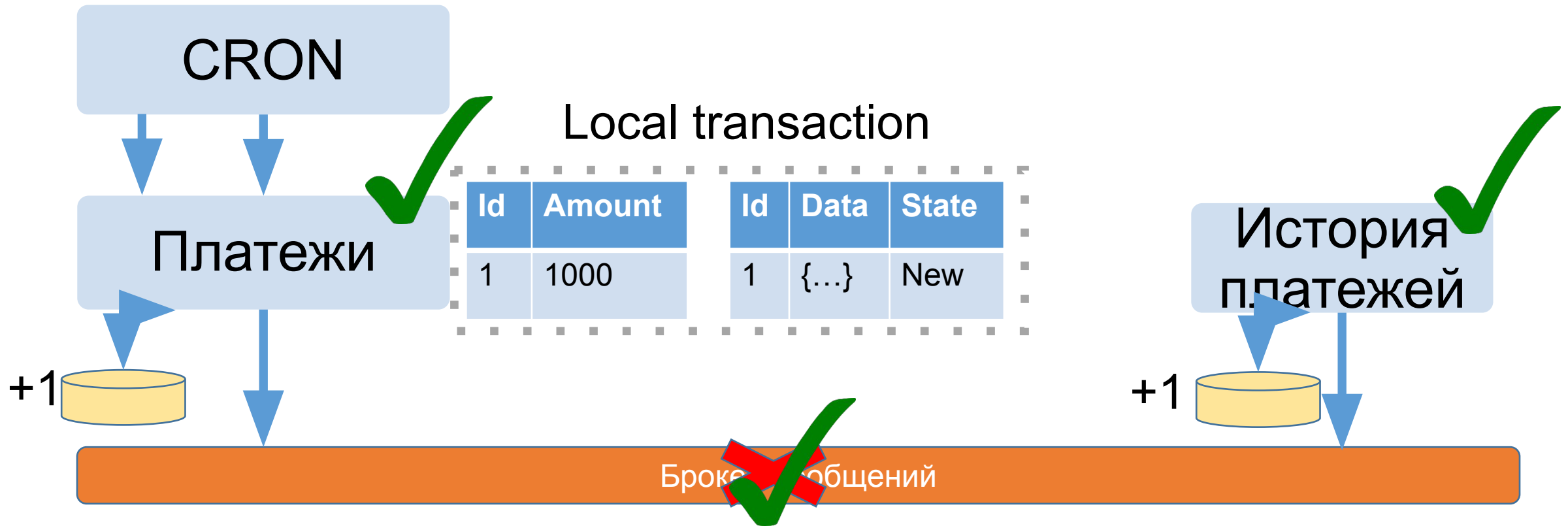


Несогласованность данных

- Нет ACID-транзакций
- Возможна несогласованность



Отложенная согласованность на событиях



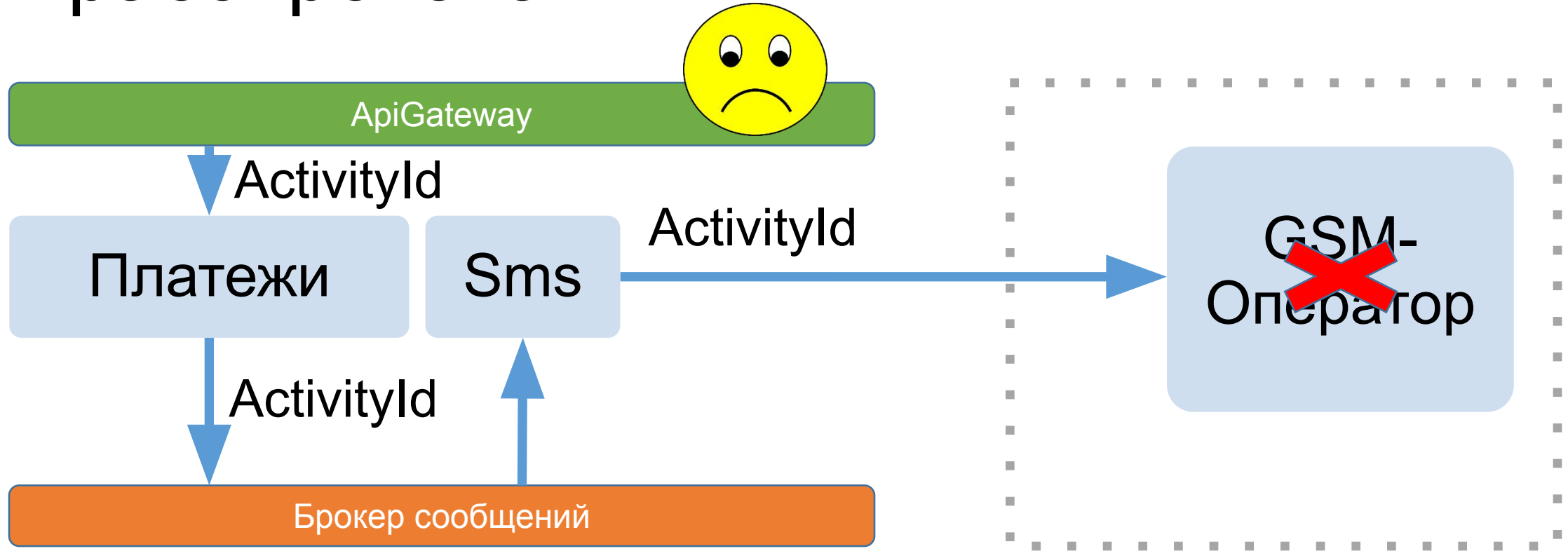
Идемпотентность

- Повторы неизбежны
- Делайте методы идемпотентными
- Регистрируйте уже принятые ID событий
- Игнорируйте более давние события

Трассировка

- Единый ID на бизнес-запрос
- Можно использовать ActivityId
- Можно воспользоваться <http://opentracing.io/>

Трассировка



ActivityId = "1234"

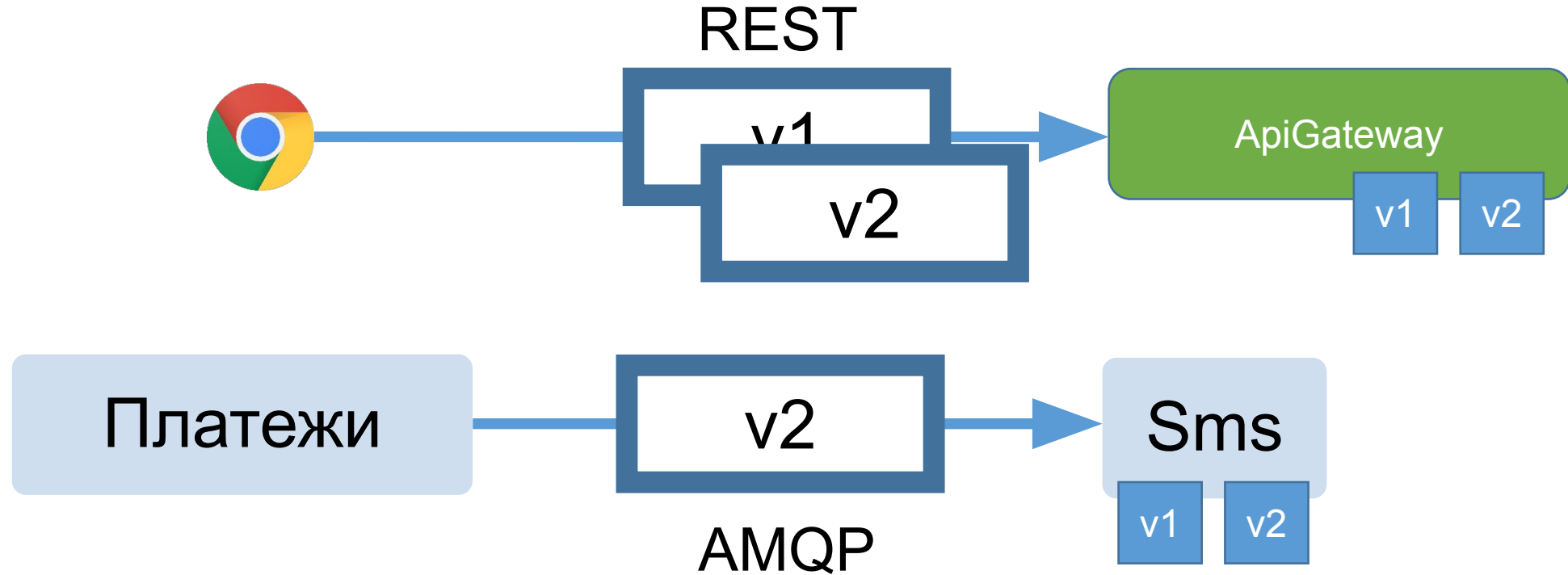
Трассировка

ActivityId	ServiceName	Logs
1234	ApiGateway	
1234	Платежи	
1234	Sms	Exception
1234	GSM Оператор	500

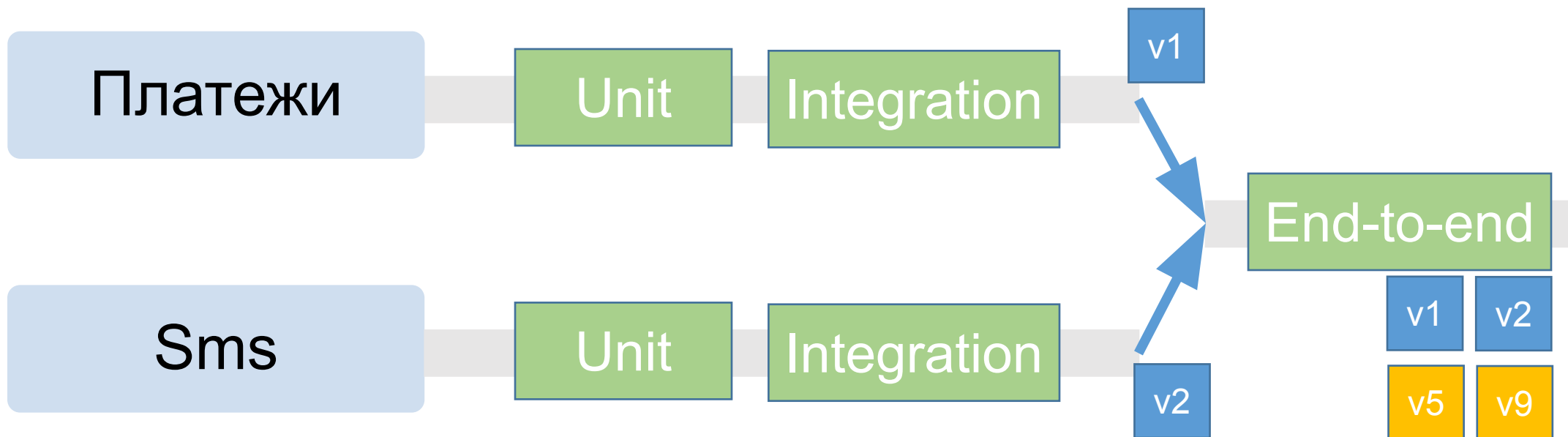
Версионирование

- Необходимо для поддержания обратной совместимости
- Позволяет развивать сервисы

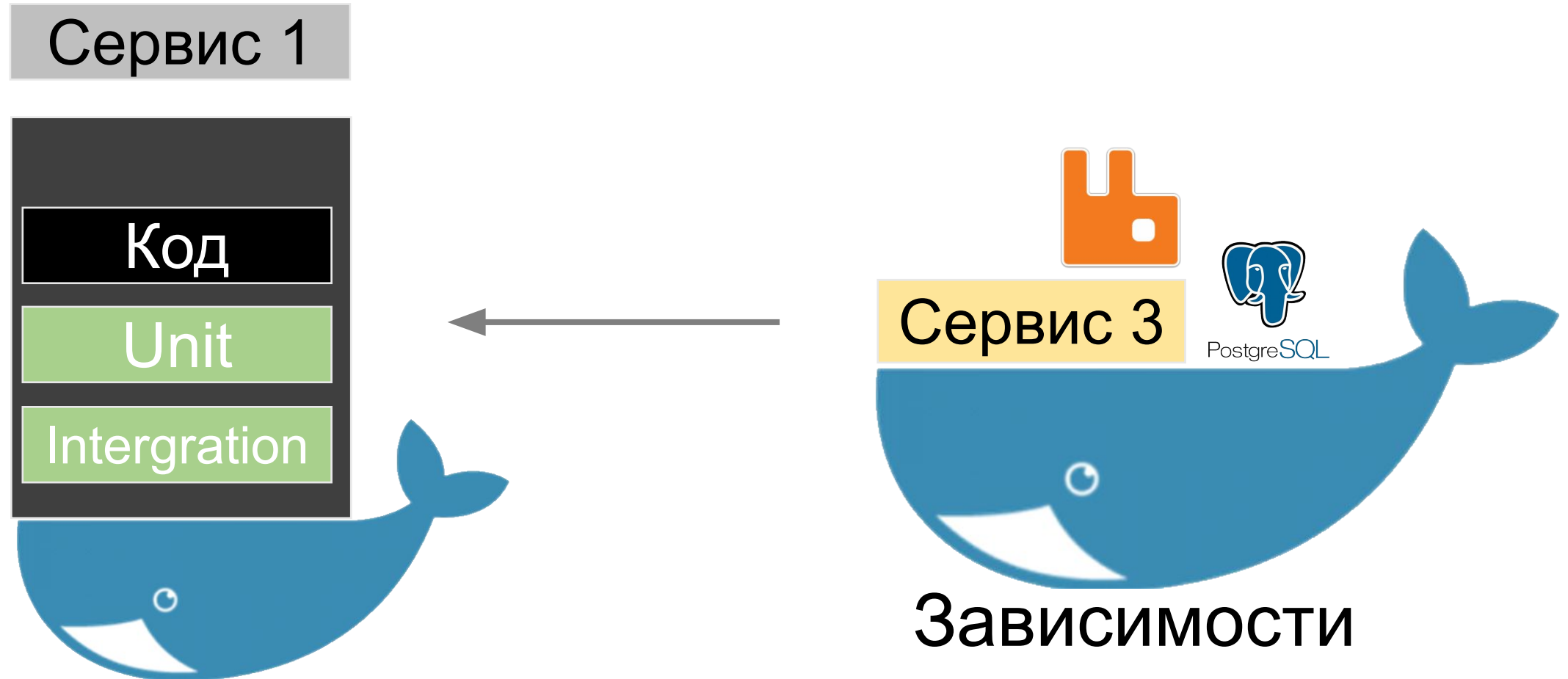
Версионирование



Тестирование



TestContainers



Развёртывание

Проблема

- Развертывать >25 сервисов
- А если с 6-кратным дублированием
- И 50 сборок в день
- А с тестированием что?
- Релизы >1 релиза в день

Решение

- DevOps
- DevOps
- И..
- Конечно, DevOps

VM or Container

VM

- Долго поднимается
- Инфраструктура настраивается
- Медленно «переезжает»



Container

- Быстро поднимается
- Инфраструктура внутри
- Легко «переезжает»

VM или (Container + Оркестрация)

VM

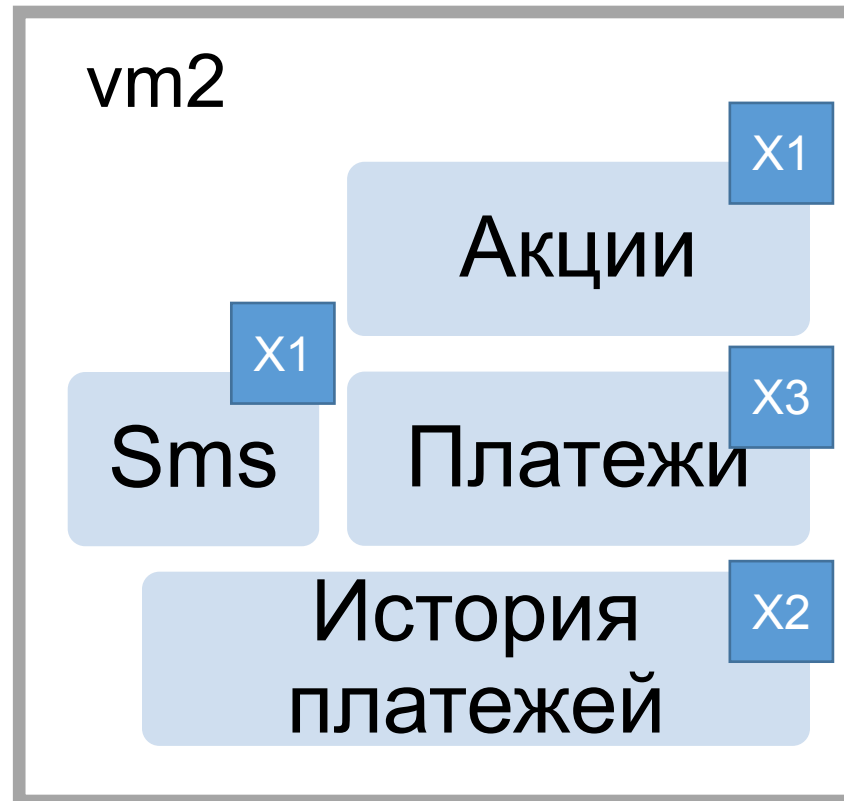
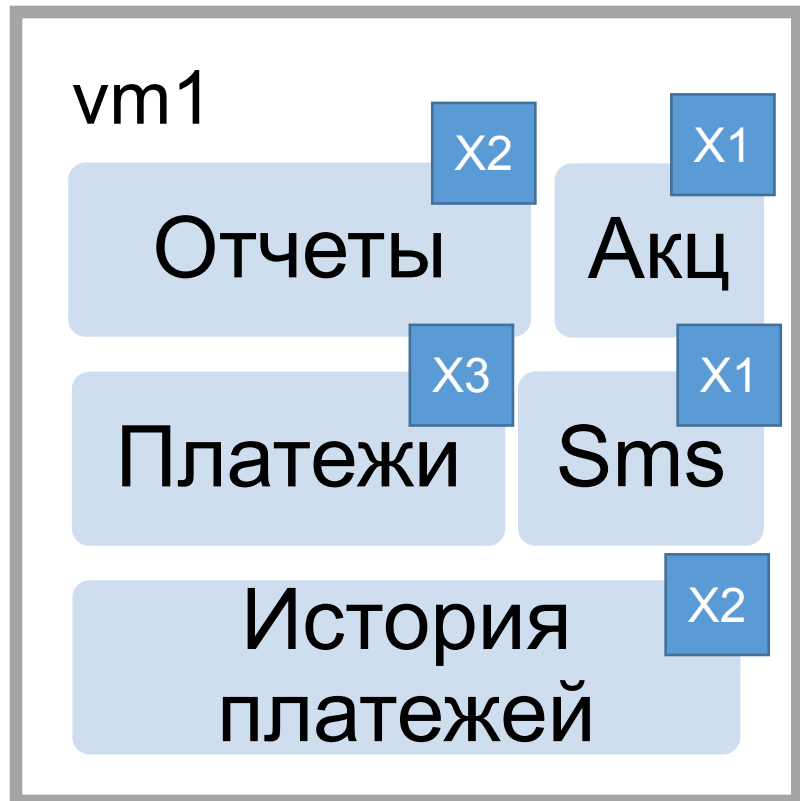
- AutoScaling – вручную
- AutoDiscovery – вручную
- Supervising – вручную
- Blue/Green deploy – вручную
- Балансировка – в ручную



Container + оркестрация (k8s)

- AutoScaling – из коробки
- AutoDiscovery – из коробки
- Supervising – из коробки
- Blue/Green deploy – из коробки
- Балансировка – из коробки*

Гибкость



16.00 – 21.00



Вопросы



Ветчинкин Кирилл

<https://www.facebook.com/k.vetchinkin>

k.vetchinkin@yandex.ru