Микросервисы

Желтак Артем

Обо мне:

- Закончил МГТУ им. Баумана по специальности инженер конструктор
- В веб разработке уже 8 долгих лет
- В GO пришел из php, возможно из-за прекрасного гофера
- Плох в создании презентаций

Виноват в следующих проектах:

Cisco Connect

HR Digital 2019
международный саммит

HR ВЕСЕННЯЯ КОНФЕРЕНЦИЯ

Обо мне:

- Закончил МГТУ им. Баумана по специальности инженер конструктор
- В веб разработке уже 8 долгих лет
- В GO пришел из php, возможно из-за прекрасного гофера
- Плох в создании презентаций

Сейчас

Обо мне:

- Закончил МГТУ им. Баумана по специальности инженер конструктор
- В веб разработке уже 8 долгих лет
- В GO пришел из php, возможно из-за прекрасного гофера
- Плох в создании презентаций

Сейчас

ПОТОК

Freadm. Будь в курсе всего.

Обо мне:

- Закончил МГТУ им. Баумана по специальности инженер конструктор
- В веб разработке уже 8 долгих лет
- В GO пришел из php, возможно из-за прекрасного гофера
- Плох в создании презентаций

Сейчас

ПОТОК

Freadm. Будь в курсе всего.

OTUS
ОНЛАЙН-ОБРАЗОВАНИЕ

План на сегодня

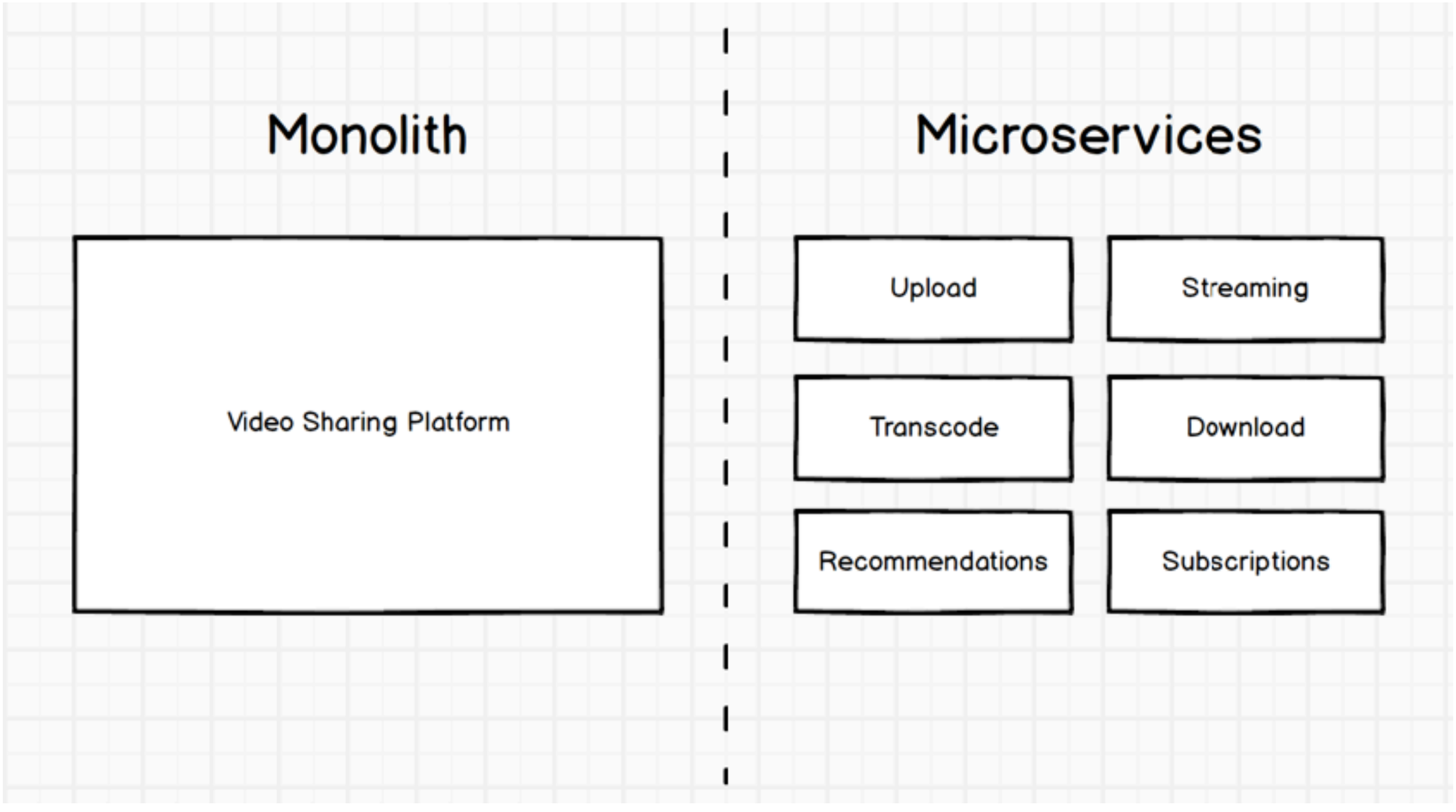
Познакомиться с микросервисной архитектурой,
рассмотреть плюсы и минусы

Разобрать подход
12 факторных приложений

Понять область применения микросервисов
и serverless архитектуры

Микросервисы

Микросервисы: отличия



Микросервисы: что за зверь?

- Микро не обязательно про размер, но про зону ответственности

Микросервисы: что за зверь?

- Микро не обязательно про размер, но про зону ответственности
- Самодостаточны, идеальны для горизонтального масштабирования

Микросервисы: что за зверь?

- Микро не обязательно про размер, но про зону ответственности
- Самодостаточны, идеальны для горизонтального масштабирования
- Разные технологии для разных задач

Микросервисы: что за зверь?

- Микро не обязательно про размер, но про зону ответственности
- Самодостаточны, идеальны для горизонтального масштабирования
- Разные технологии для разных задач
- Распределенная кодовая база

Микросервисы: плюсы

- Независимая разработка

Микросервисы: плюсы

- Независимая разработка
- Независимое развёртывание

Микросервисы: плюсы

- Независимая разработка
- Независимое развёртывание
- Независимая масштабируемость

Микросервисы: плюсы

- Независимая разработка
- Независимое развёртывание
- Независимая масштабируемость
- Повторное использование

Микросервисы: плюсы

- Независимая разработка
- Независимое развёртывание
- Независимая масштабируемость
- Повторное использование
- Можно переписать,
  а не доработать

Микросервисы: плюсы

- Независимая разработка
- Независимое развёртывание
- Независимая масштабируемость
- Повторное использование
- Можно переписать,
  а не доработать
- Проще тестирование

Микросервисы: минусы

- Возросшая сложность для тестирования

Микросервисы: минусы

- Возросшая сложность для тестирования
- Возросшая сложность для разработчиков

Микросервисы: минусы

- Возросшая сложность для тестирования
- Возросшая сложность для разработчиков
- Возросшая сложность для эксплуатации/Devops

Микросервисы: минусы

- Возросшая сложность для тестирования
- Возросшая сложность для разработчиков
- Возросшая сложность для эксплуатации/Devops
- Выше требования к компетенции

Микросервисы: ирония

2014 — нужно внедрить микросервисы для решения проблем с монолитами.
2016 — нужно внедрить Docker, чтобы решить проблемы с микросервисами.
2018 — нужно внедрить Kubernetes, чтобы решить проблемы с Docker.

Микросервисы: ирония

2014 — нужно внедрить микросервисы для решения проблем с монолитами.
2016 — нужно внедрить Docker, чтобы решить проблемы с микросервисами.
2018 — нужно внедрить Kubernetes, чтобы решить проблемы с Docker.
2020 — To be continiued

Микросервисы: минусы часть 2

- Удаленные вызовы дороже локального исполнения

Микросервисы: минусы часть 2

- Удаленные вызовы дороже локального исполнения
- Реальные системы обычно не имеют чётко определённых границ

Микросервисы: минусы часть 2

- Удаленные вызовы дороже локального исполнения
- Реальные системы обычно не имеют чётко определённых границ
- Сложности stateful

Микросервисы: минусы часть 2

- Удаленные вызовы дороже локального исполнения
- Реальные системы обычно не имеют чётко определённых границ
- Сложности stateful
- Версионность и работа с ней

Микросервисы: минусы часть 2

- Удаленные вызовы дороже локального исполнения
- Реальные системы обычно не имеют чётко определённых границ
- Сложности stateful
- Версионность и работа с ней
- Сложности взаимодействия между сервисами

Микросервисы: минусы часть 2

- Удаленные вызовы дороже локального исполнения
- Реальные системы обычно не имеют чётко определённых границ
- Сложности stateful
- Версионность и работа с ней
- Сложности взаимодействия между сервисами
- Распределенные транзакции

Микросервисы: минусы часть 2

- Удаленные вызовы дороже локального исполнения
- Реальные системы обычно не имеют чётко определённых границ
- Сложности stateful
- Версионность и работа с ней
- Сложности взаимодействия между сервисами
- Распределенные транзакции
- Попытка замаскировать монолит

Микросервисы: переходить?

- Размер команды

Микросервисы: переходить?

- Размер команды
- Stateless

Микросервисы: переходить?

- Размер команды
- Stateless
- У вас одно приложение или несколько

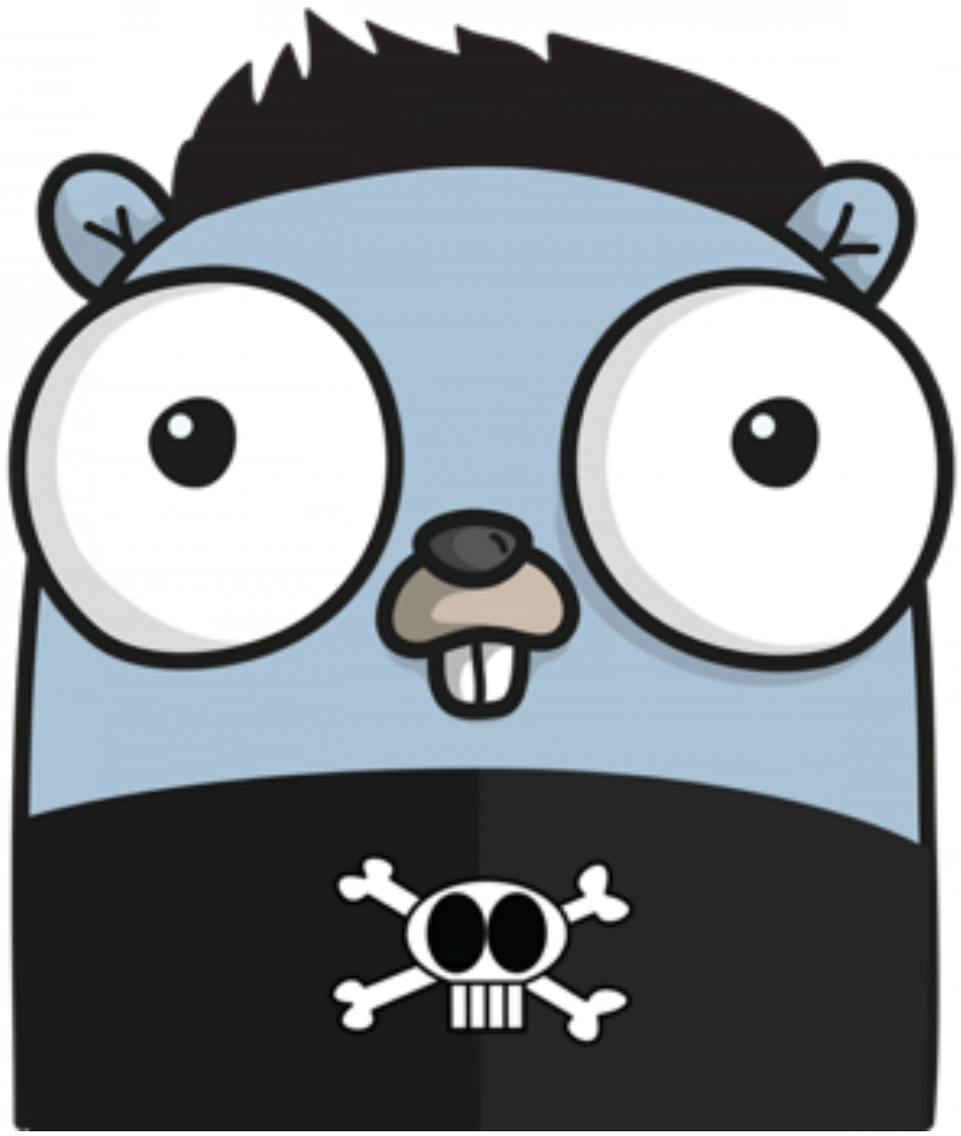
Микросервисы: переходить?

- Размер команды
- Stateless
- У вас одно приложение или несколько
- Монолитные зависимости и границы сервисов

Микросервисы: переходить?

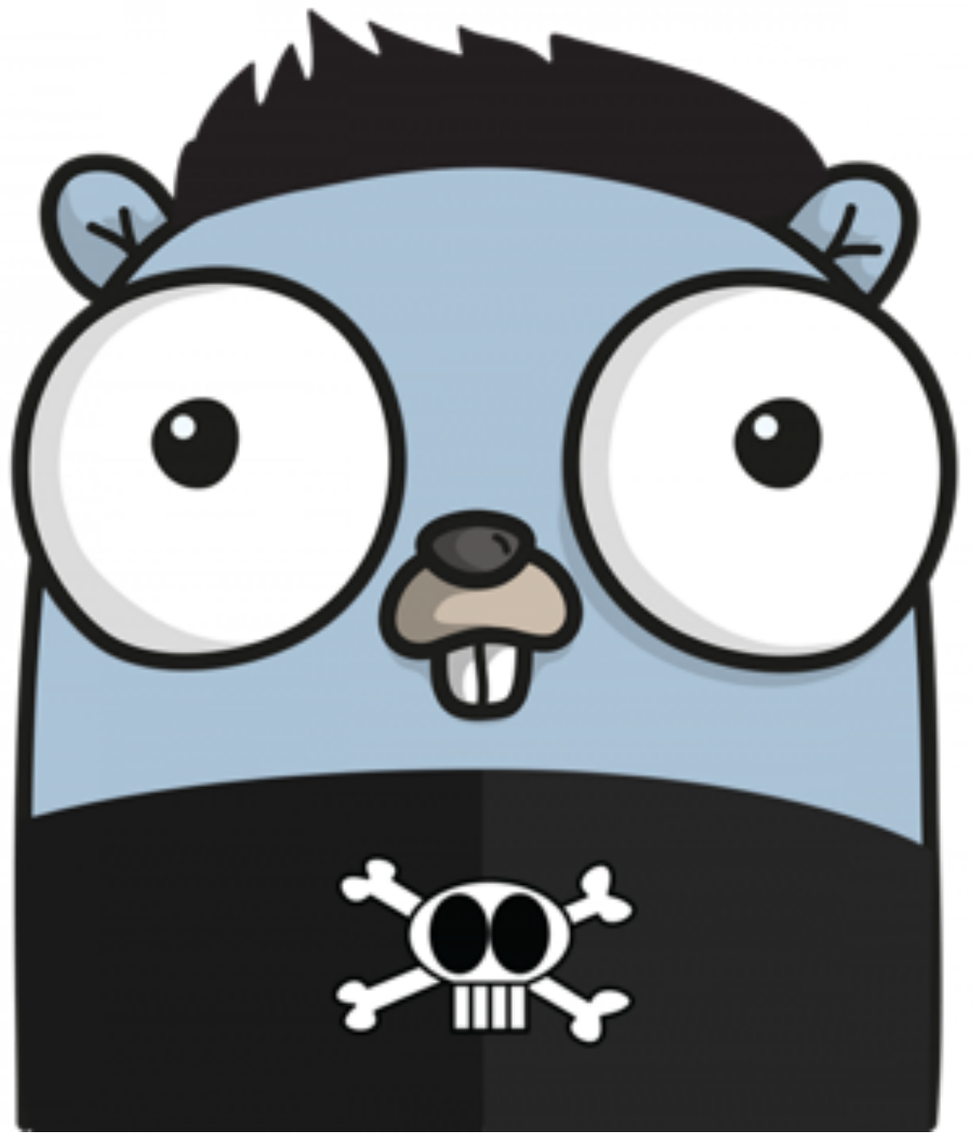
- Размер команды
- Stateless
- У вас одно приложение или несколько
- Монолитные зависимости и границы сервисов
- Есть ли компетенция на уровне DevOps

12 факторные приложения



12 факторные приложения

1. Кодовая база
Одна кодовая база, отслеживаемая в системе контроля версий, – множество развёртываний

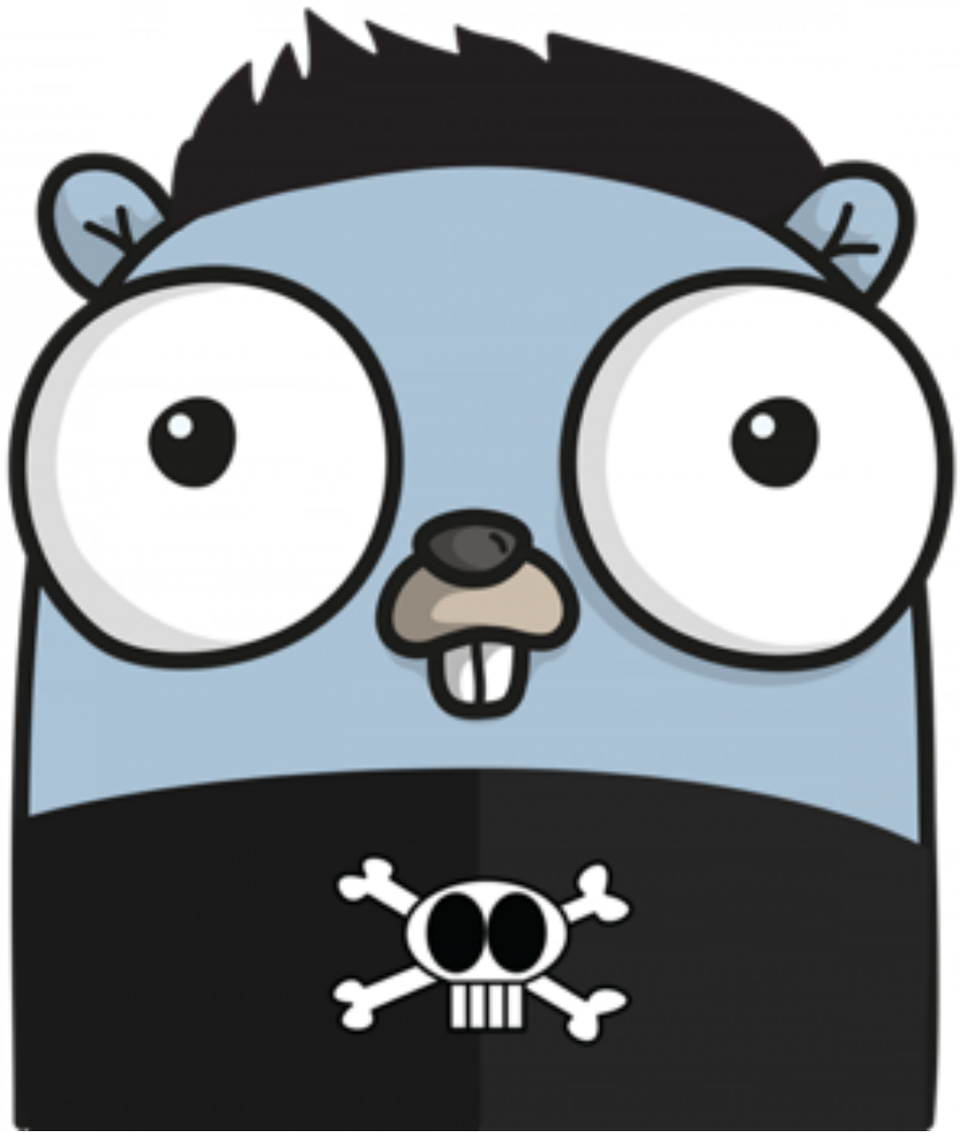


12 факторные приложения

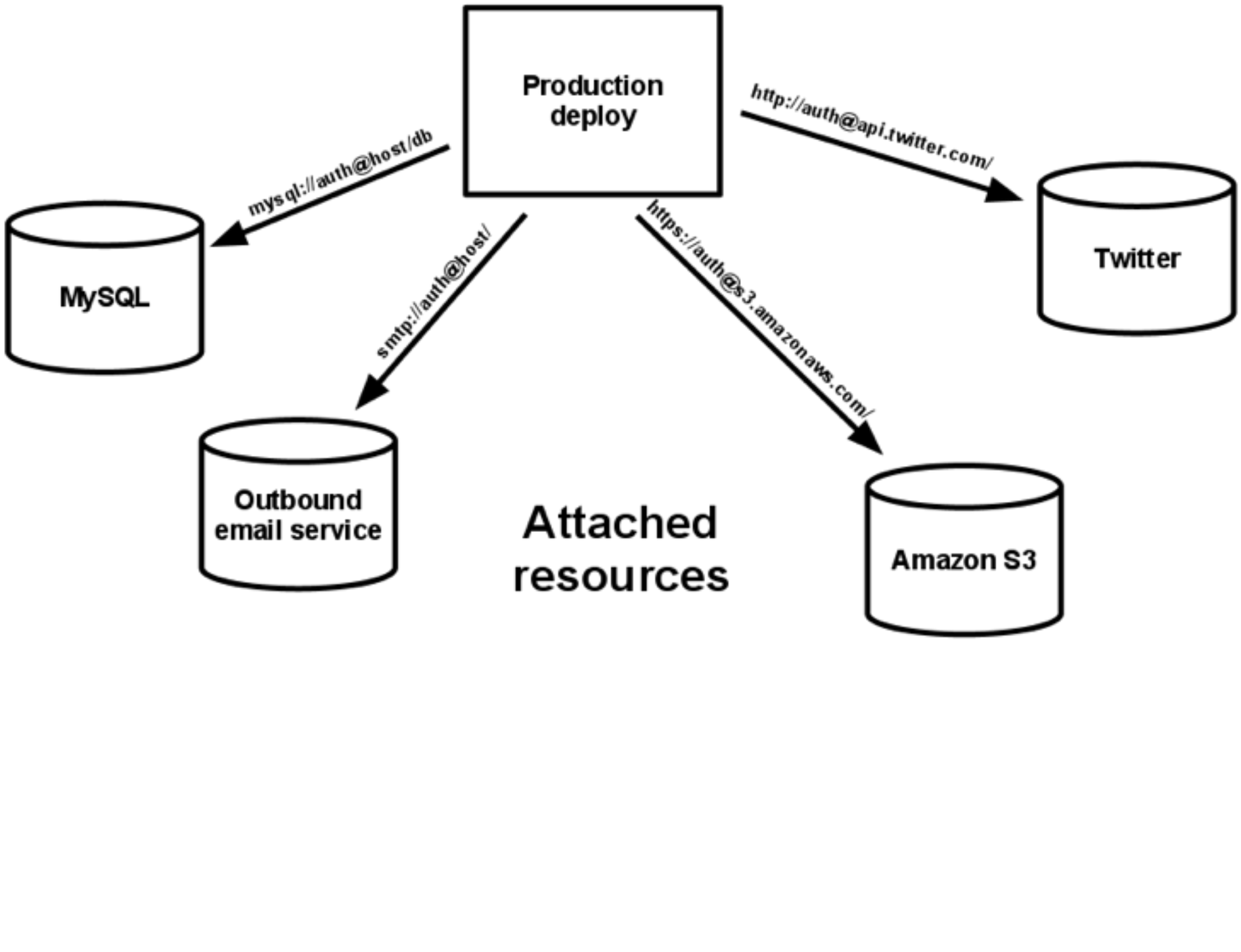
1. Кодовая база
Одна кодовая база, отслеживаемая в системе контроля версий, – множество развёртываний
2. Работа с зависимостями
Явно объявляйте и изолируйте зависимости

12 факторные приложения

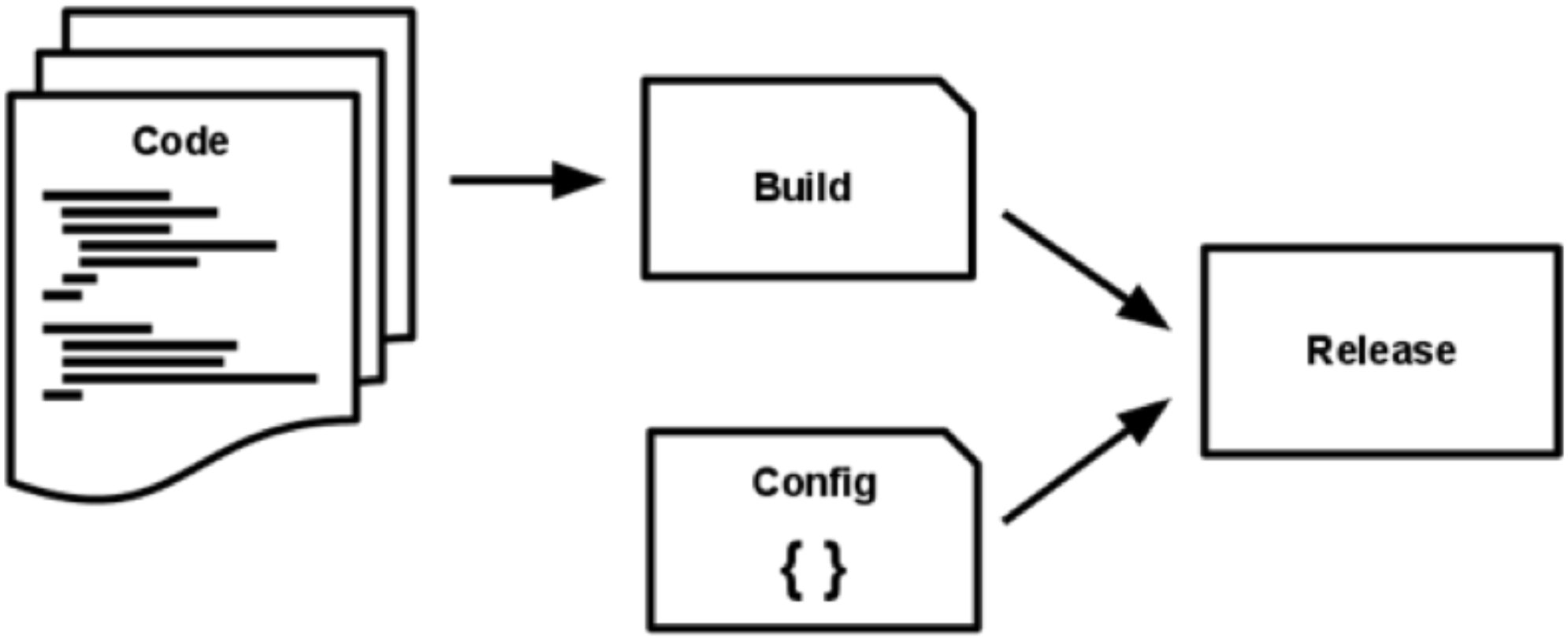
1. Кодовая база
Одна кодовая база, отслеживаемая в системе контроля версий, – множество развёртываний
2. Работа с зависимостями
Явно объявляйте и изолируйте зависимости
3. Конфигурация
Сохраняйте конфигурацию в среде выполнения



4. Сторонние службы



5. Сборка, релиз, выполнение



12 факторные приложения

6. Процессы
Запускайте приложение как один или несколько процессов не сохраняющих внутреннее состояние (stateless)

12 факторные приложения

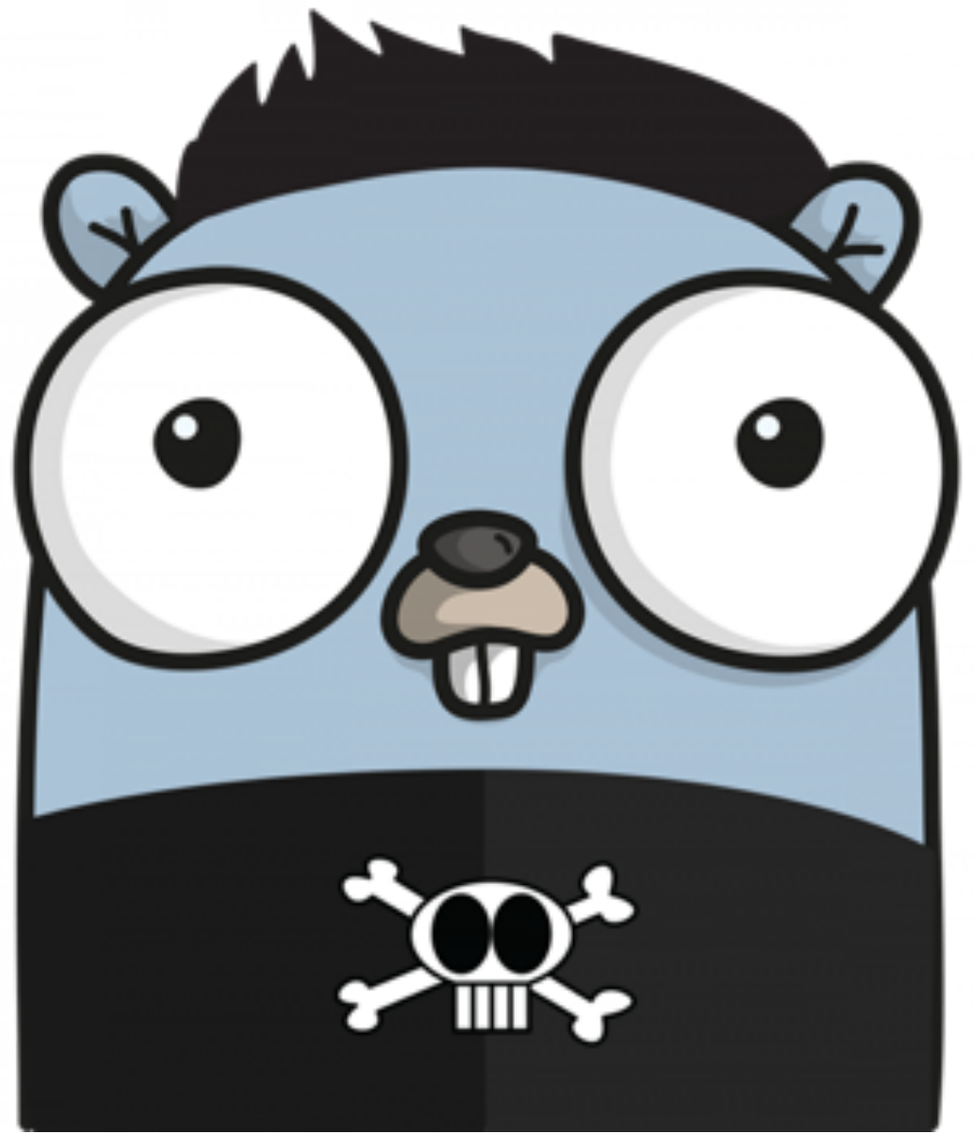
6. Процессы
Запускайте приложение как один или несколько процессов не сохраняющих внутреннее состояние (stateless)
7. Привязка портов
Экспортируйте сервисы через привязку портов

12 факторные приложения

6. Процессы
Запускайте приложение как один или несколько процессов не сохраняющих внутреннее состояние (stateless)
7. Привязка портов
Экспортируйте сервисы через привязку портов
8. Параллелизм
Масштабируйте приложение с помощью процессов

12 факторные приложения

9. Утилизируемость
Максимизируйте надёжность с помощью быстрого запуска и корректного завершения работы



12 факторные приложения

9. Утилизируемость
Максимизируйте надёжность с помощью быстрого запуска и корректного завершения работы

10. Паритет разработки/работы приложения
Держите окружения разработки, промежуточного развёртывания (staging) и рабочего развёртывания (production) максимально похожими

12 факторные приложения

9. Утилизируемость
Максимизируйте надёжность с помощью быстрого запуска и корректного завершения работы
10. Паритет разработки/работы приложения
Держите окружения разработки, промежуточного развёртывания (staging) и рабочего развёртывания (production) максимально похожими
11. Журналирование
Рассматривайте журнал как поток событий

12 задачи администрирования

- Разовые процессы администрирования (миграции, исправления базы) должны подчиняться тем же правилам, что и остальные процессы: Код задач должен лежать в репозитории, чтобы соответствовать коду основного приложения
- Зависимости должны быть объявлены в основном манифесте зависимостей, чтобы процесс мог быть выполнен при обычном развертывании
- Конфигурация задачи должна находиться в переменных окружения, чтобы ее можно было выполнить в разных окружениях

Serverless подход

- Сервер все таки есть

Serverless подход

- Сервер все таки есть
- Function as Service

Serverless подход

- Сервер все таки есть
- Function as Service
- Долгий запуск,
  короткое время жизни

Serverless подход

- Сервер все таки есть
- Function as Service
- Долгий запуск,
  короткое время жизни
- Нет зависимости от инфраструктуры,
  но есть vendor lock

Serverless подход

Событие	Действие, которое выполняет функция
В хранилище загрузили картинку товара	Сжать картинку и выгрузить в каталог
В базе данных обновился адрес физического магазина	Подгрузить в карты новое местоположение
Клиент оплачивает товар	Запустить обработку платежа

Serverless подход

- Сократить простой ресурсов

Serverless подход

- Сократить простой ресурсов
- Ускорить разработку (писать нужно только бизнес логику)

Serverless подход

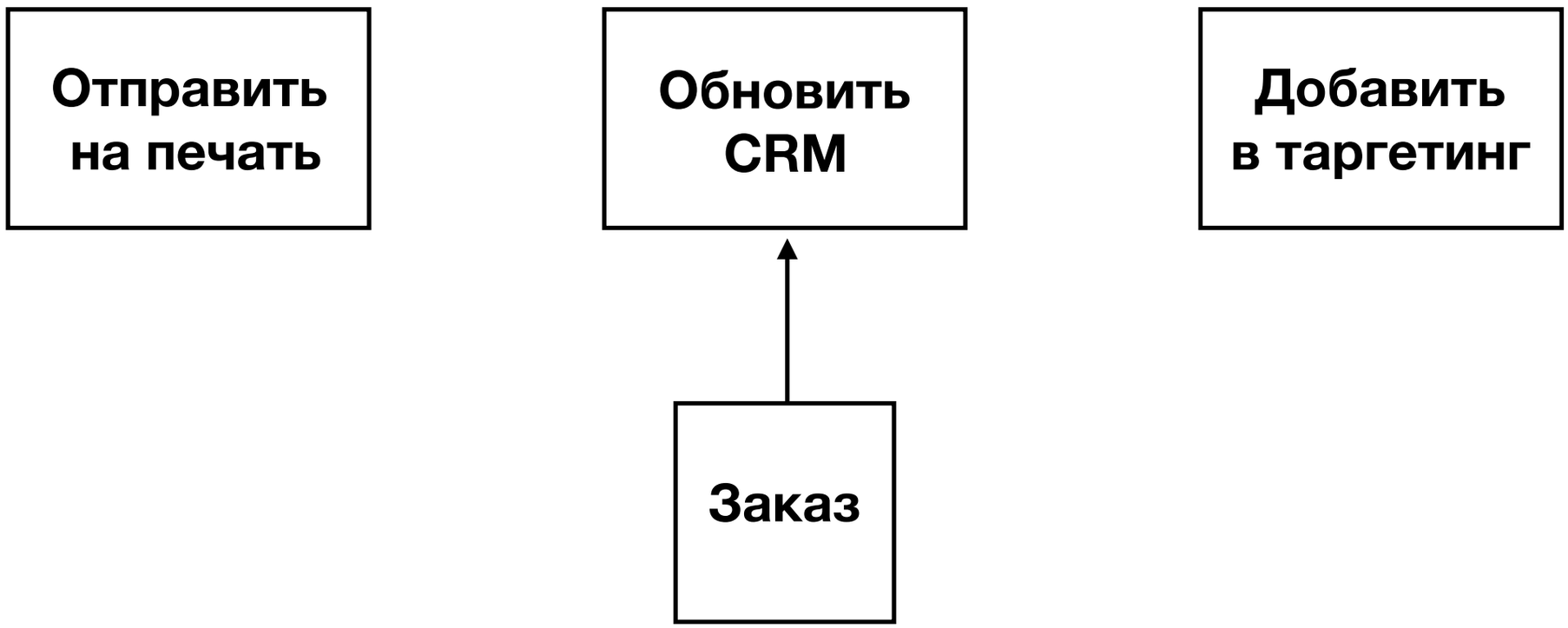
- Сократить простой ресурсов
- Ускорить разработку (писать нужно только бизнес логику)
- Масштабирование никогда не было таким простым

Serverless подход

- Сократить простой ресурсов
- Ускорить разработку (писать нужно только бизнес логику)
- Масштабирование никогда не было таким простым
- Не подходит для realtime систем

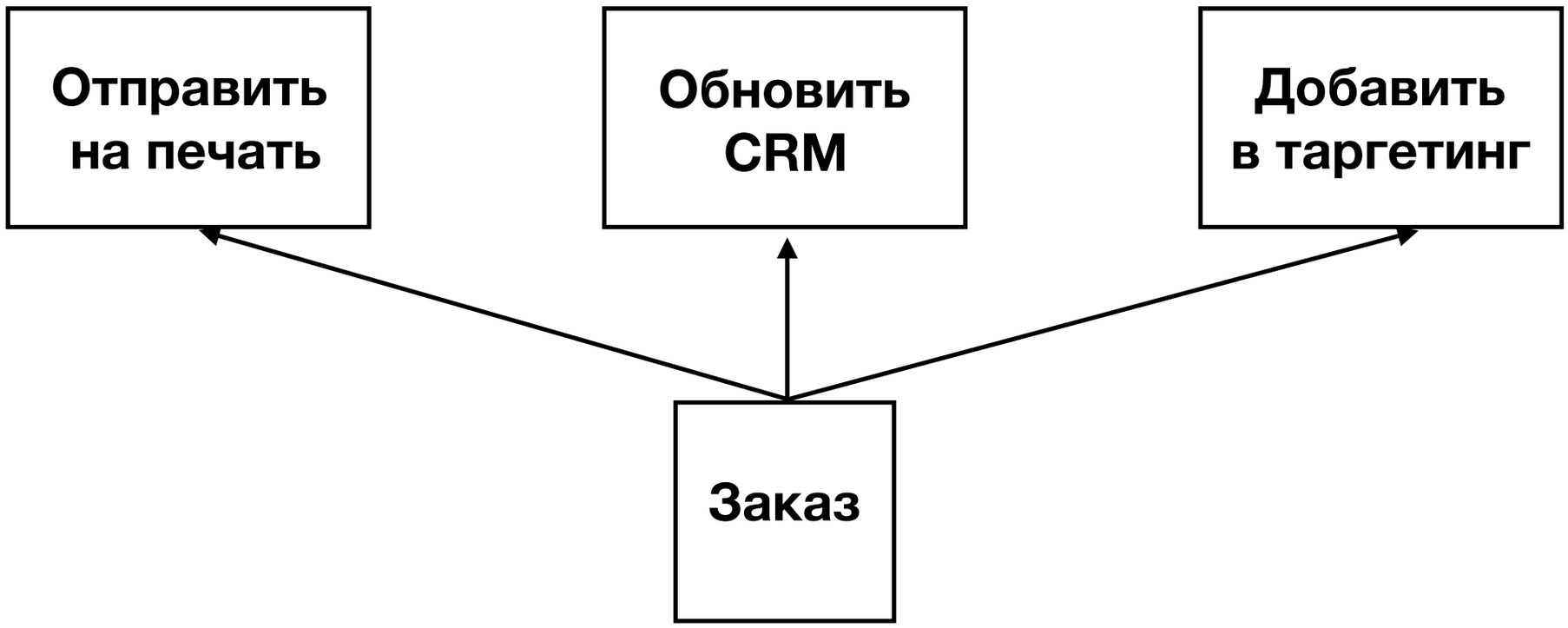
Event driven: Бизнес подход

Event driven: Бизнес подход



Event driven: Бизнес подход

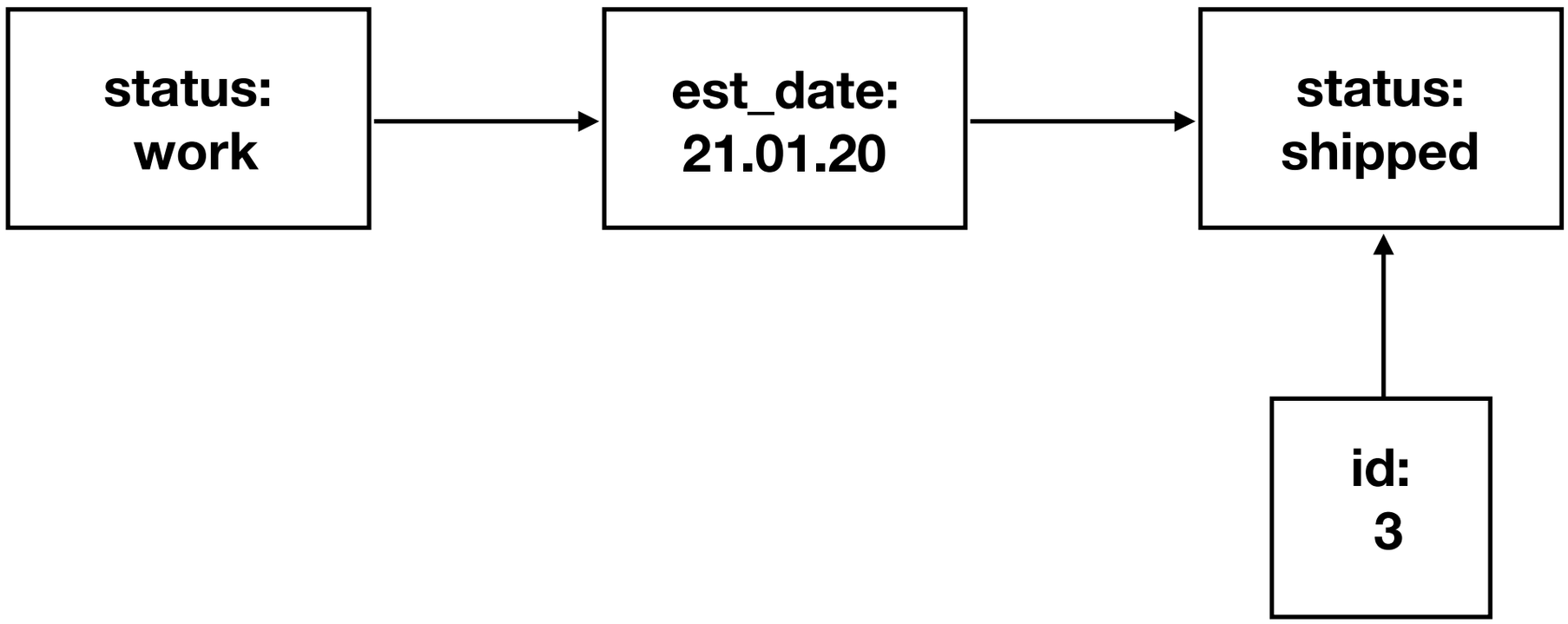
Event driven: Бизнес подход



Event driven: Developer подход

- Отправка в брокер сообщений из кода
- Чтение бинлога и отправка в брокер
- Вспомогательная таблица из которой отправляются события в брокер

Event driven: Изменение данных



Command Query Responsibility Segregation (CQRS)

- Read only
- Материализованное представление данных
- Можно использовать для snapshot’ов
- Позволяет делать выборки

Практическая часть

Оценка решения

- Простота
- Экономия ресурсов
- Удобство для пользователя

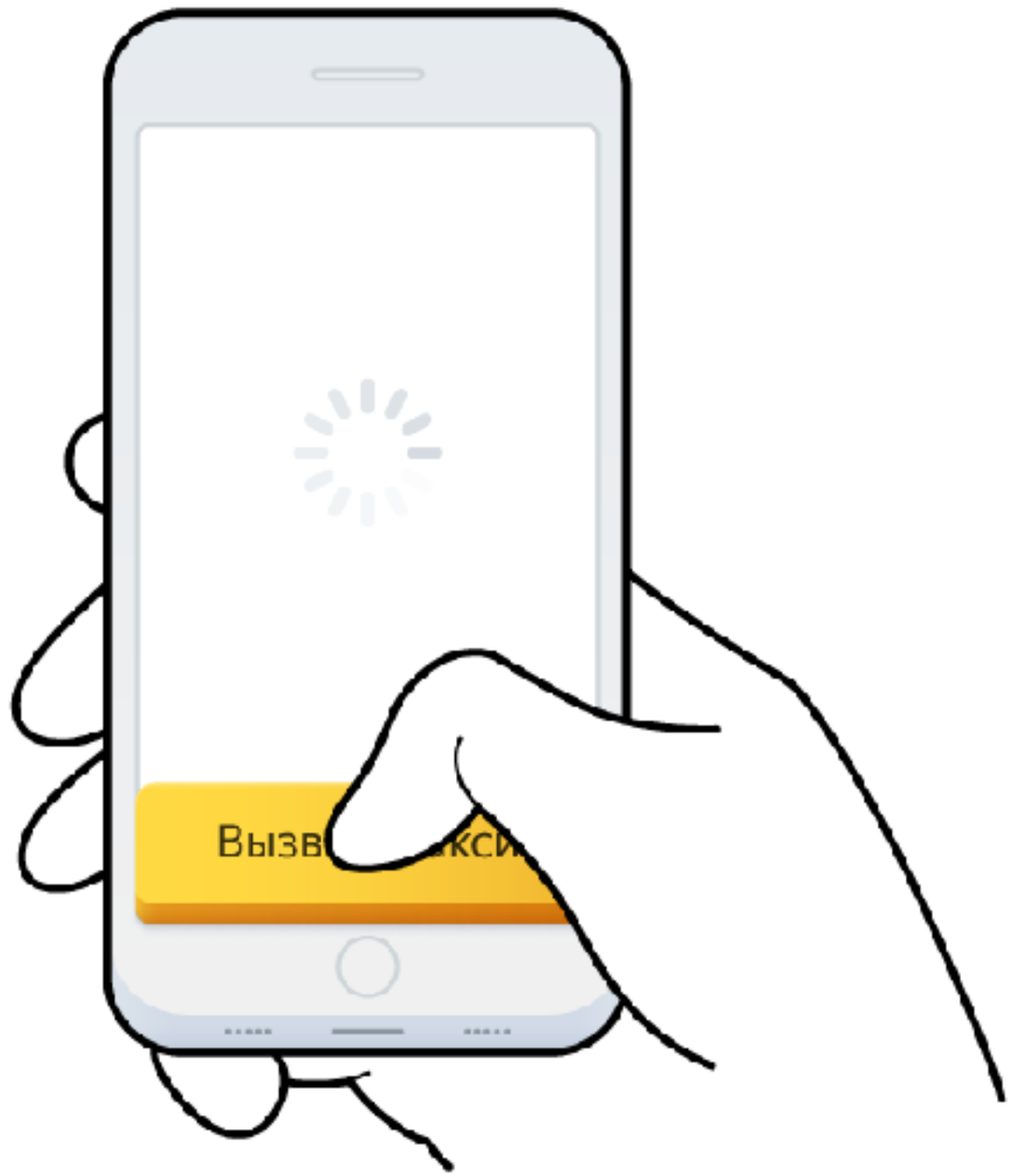
Практическая часть

Пути констистентности

- Путь надежды
- Двух фазные комиты
- Отложенный путь
- Привязка к определенному шарду по критерию

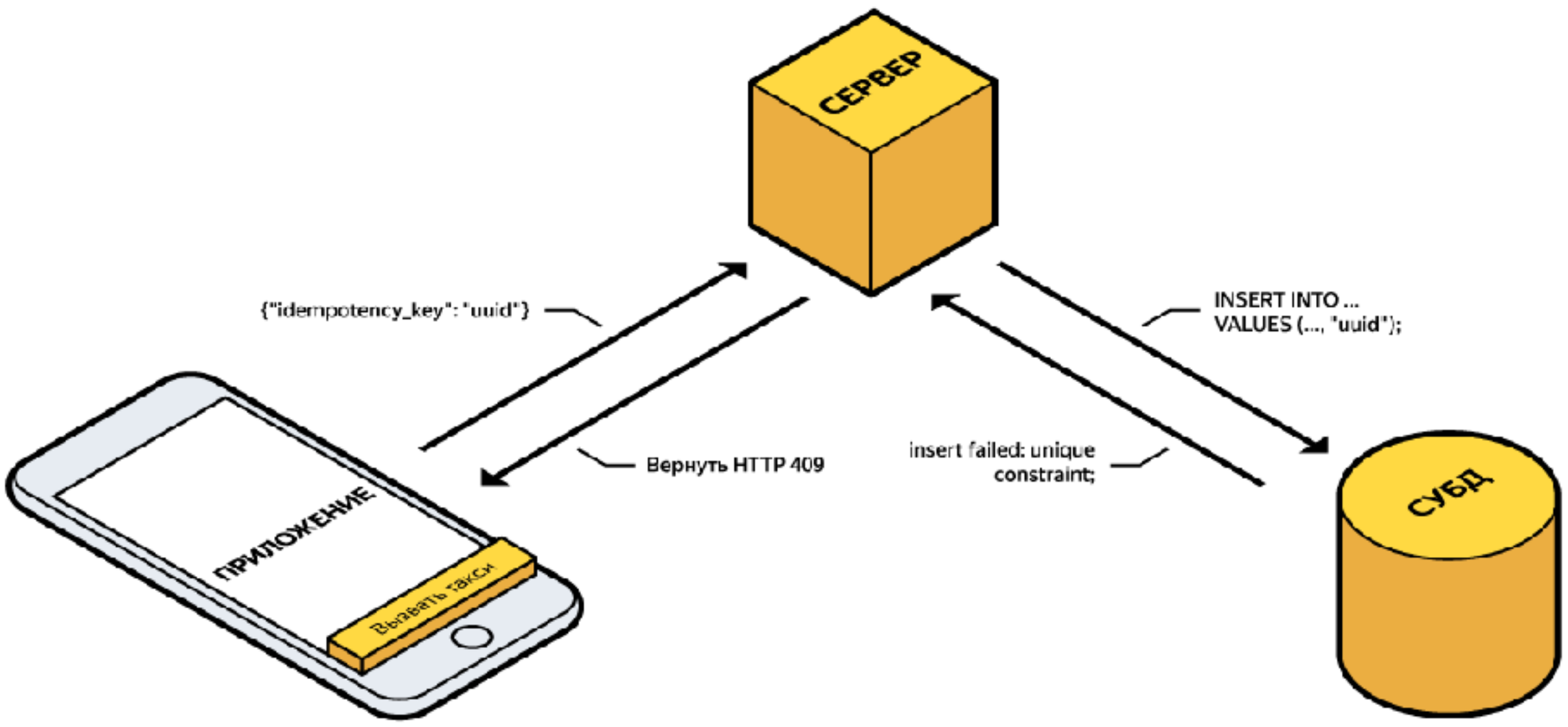
- Легко, дешево, пользователи не ок
- Сложно, ресурсов много, пользователи ок
- Легко, когда нибудь ок, пользователи +-
- Легко, ок, пользователи не ок

История заказа такси



По мотивам https://habr.com/ru/company/yandex/blog/442762/

История заказа такси



Минутка кода

```
func doWork(new Payment) {
    db.BeginTransaction()
    err := procced(new)
    if err == nil {
        err = db.Commit()
        if err == nil {
            sendQueue(EmailEvent(new))
        }
    } else {
        db.Rollback()
    }
}
```

```
func doWorkAlt(new Payment) {
    db.BeginTransaction()
    err := procced(new)
    sendQueue(EmailEvent(new))
    if err == nil {
        err = db.Commit()
    } else {
        db.Rollback()
    }
}
```

Минутка кода

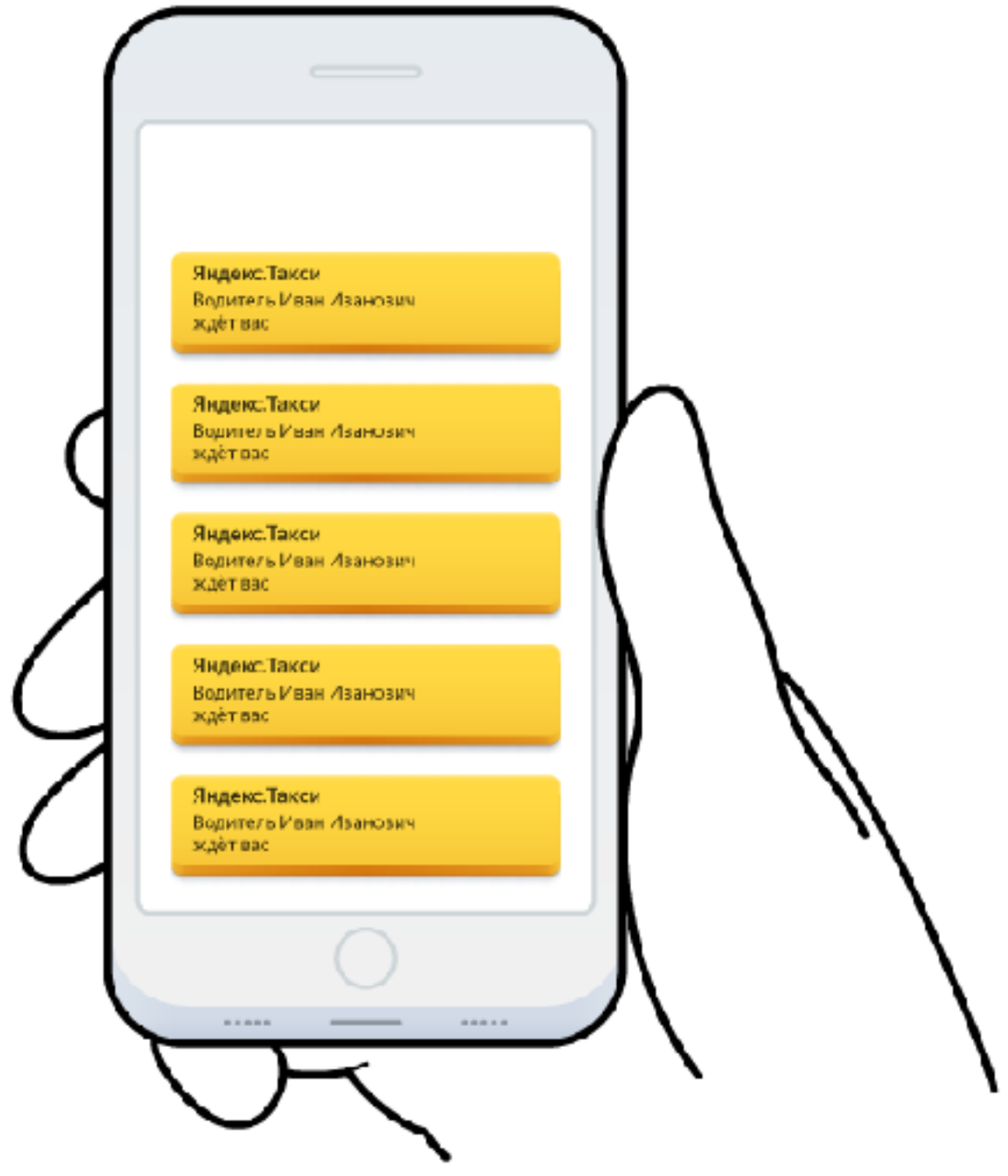
- Deferable events - каждый ивент знает, что будет дальше - матрешка
- Вспомогательные обертки. По названию определяется выполнение event’a сейчас или по commit транзакции
- Страховочные скрипты, которые проверяли факт выполнения
- Ошибочные очереди

Подводные камни

- Полная информация / отправка id
- Абсолютные значения (25 лет -> 26 лет, а не +1 год)
- Системы очередей FIFO - дают гарантию отправки по порядку,
  но не по выполнению
- Коварное время, на разных машинах

Идемпотентность при внешних операциях

- at least once delivery
- at most once delivery



Рекомендую


O'REILLY
ВЫСОКО-
НАГРУЖЕННЫЕ
ПРИЛОЖЕНИЯ
Программирование
масштабирование
поддержка
ПИТЕР
Мартин Клеппман

Моя история

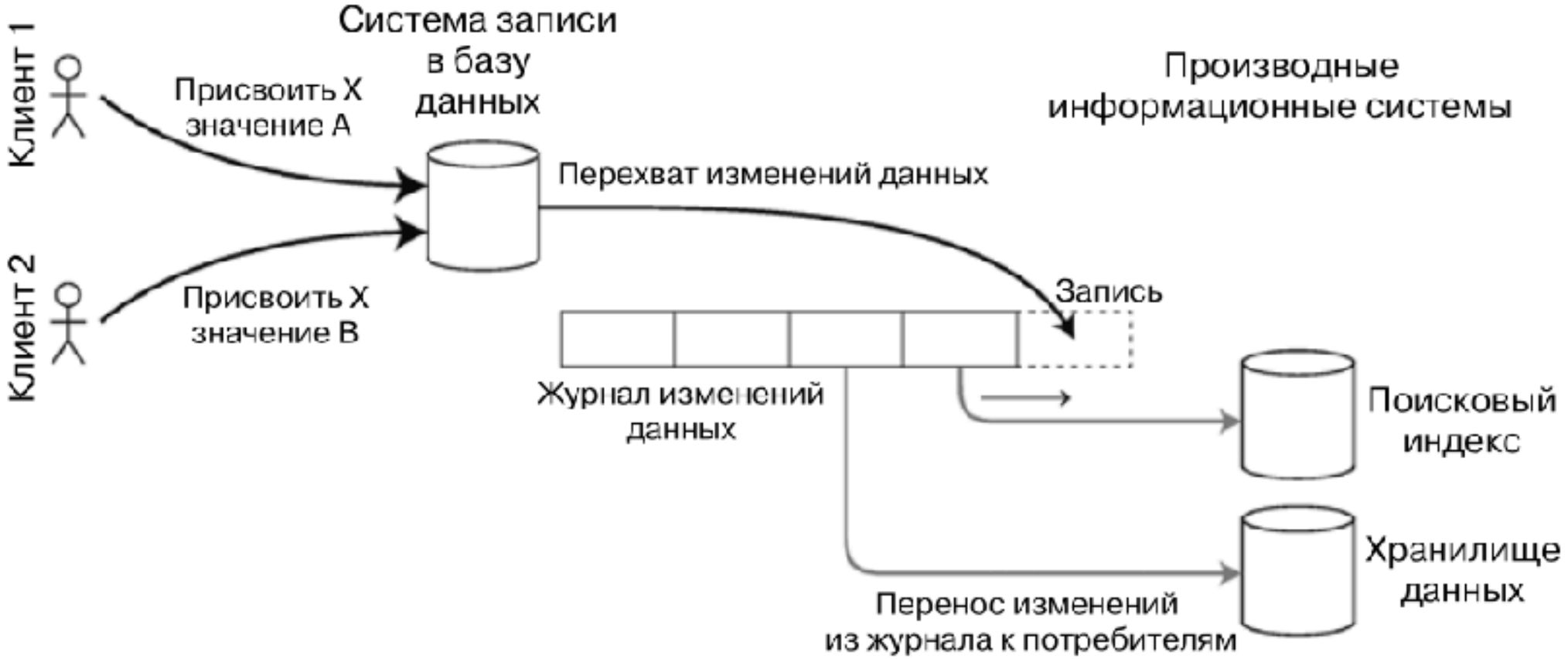


Рис. 11.5. Данные принимаются в том порядке, в каком они были записаны в базу, а затем изменения в том же порядке переносятся на другие системы

Моя история

- Эксплуатация довольна, используем старый механизм репликации
- Данные попадают после транзакции
- Не учитывается откат изменений - удаление записи придет как новая запись в бинлоге
- Есть отставания

Спасибо за внимание.
Вопросы?

Желтак Артем