# CQRS & ES для онлайн-аукционов

~~Слабоумие и отвага~~

Грабли и опыт

# О докладчике

Михаил Селиверстов, работаю в Аркадии с 2014 года, преимущественно со стеком .NET

bratmo@gmail.com

# Как все начиналось

Дело было так…

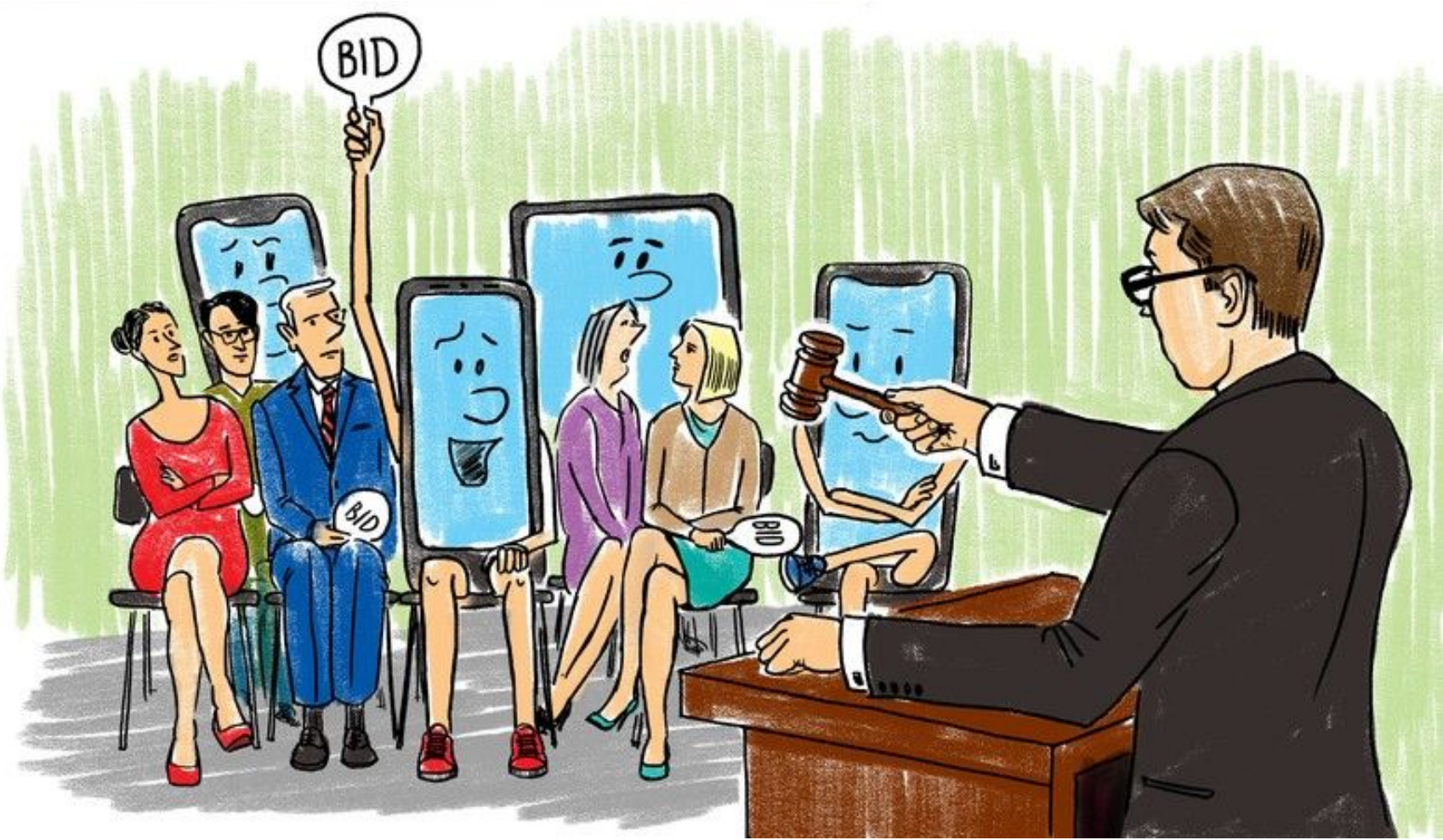
Пришел заказчик

# Аукцион - это...

- Лоты
- Покупатели
- Ставки

# Timed-аукцион - это...

# Live-аукцион - это...


BID
BID
BID

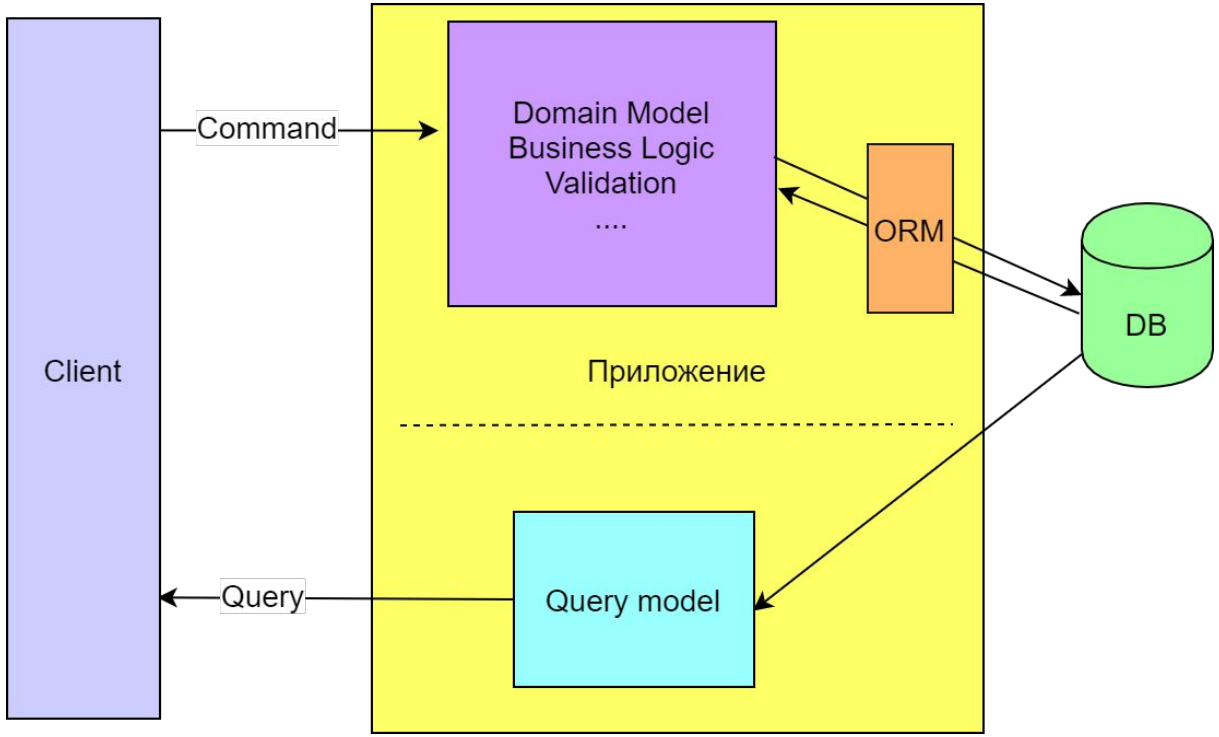
# Специфика live-аукционов

- Обработка запросов в реальном времени
- Интерфейс не должен подвисать никогда
- Множество одновременных действий над одним объектом
- Учет всех входящих запросов
- Разрешение спорных ситуаций - нужна реальная история
- Масштабирование

# Почему CQRS?
# Почему Event Sourcing?

# CQRS - это...

Command
Query
Responsibility
Segregation




- Command меняет state
- Query не меняет state

# Event Sourcing - это...

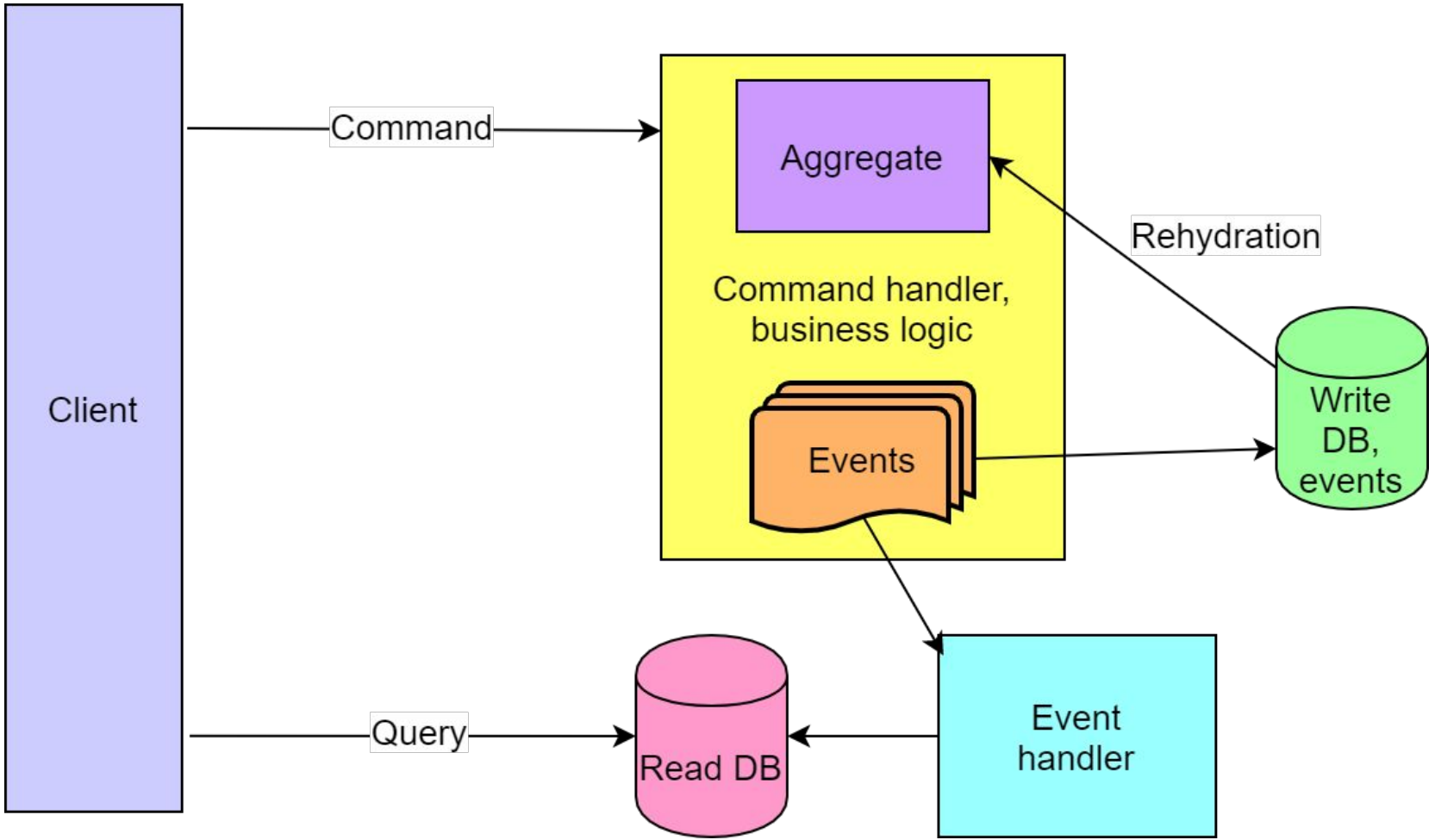
- Хранятся не сами данные, а события
- События - результат обработки команды бизнес-логикой
- Одна команда - много событий

- Пример команды - сделать ставку
- Пример событий команды - сделана ставка, снят статус FairWarning

# CQRS & Event Sourcing. Агрегаты

- Доменная модель - агрегаты
- Агрегат имеет версию
- Каждое событие инкрементирует версию агрегата
- Команда применяется к агрегату последней версии
- Восстановление агрегата = регидратация
- Регидратация агрегата = применение всех событий агрегата к начальному состоянию
- Снапшоты - “снимки” агрегата определенной версии

- Пример агрегата - лот


Client
Command
Aggregate
Command handler,
business logic
Events
Rehydration
Write
DB,
events
Query
Read DB
Event
handler

# Так почему все-таки CQRS & Event Sourcing

- История. События - source of truth
- События - для push-нотификаций

# Реализация
# Тонкости и сложности

# Выбор фреймворка

Технологический стек:

- .NET, C#
- Microsoft SQL
- Azure

Фреймворк Chinchilla:

+ Azure Service Bus
+ SignalR web sockets

# Зачем нужен фреймворк

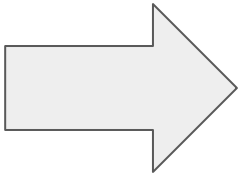
Фреймворк решает:

- Агрегат, команда, событие, версионирование агрегатов, регидратация, снапшоты
- Интерфейсы к СУБД. Сохранение/загрузка событий и снапшотов агрегатов
- Интерфейсы к очередям — запись и чтение команд и событий
- Интерфейс для работы с веб-сокетами

# Выбор агрегатов

**Бизнес-сущности:**

- Аукцион
- Лот
- Ставка



**Агрегаты:**

```
public class Auction
{
    public AuctionState State { get; private set; }
    public Guid? CurrentLotId { get; private set; }
    public List<Guid> Lots { get; }
}

public class Lot
{
    public Guid? AuctionId { get; private set; }
    public LotState State { get; private set; }
    public decimal NextBid { get; private set; }
    public Stack<Bid> Bids { get; }
}
```

# Выбор агрегатов. Анализ

**Получилось неудачно!**

Пример: аукцион на паузе

# Выбор агрегатов. Альтернативы?

Один “большой” агрегат вместо двух “небольших”:

- будет “тяжелым”
- больше будет конкурентности при отправке ему команд

Пример: ставки на текущий лот, ставки absentee bids на следующие лоты, отмена absentee bids, управление аукционом

# Выбор агрегатов. Workaround

Как же выкрутиться?

- Выполнять две команды на одно действие пользователя
- Делать изменения в двух агрегатах в рамках одной команды

# Выбор агрегатов

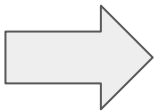
**Вывод**:
внимательнее при разделении на агрегаты!

# Параллельная обработка команд

Обработка команд может вестись параллельно. Нужно это вообще?

Команда 1

Команда 2



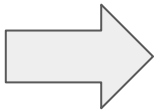
Агрегат 1

Агрегат 2

# Параллельная обработка команд

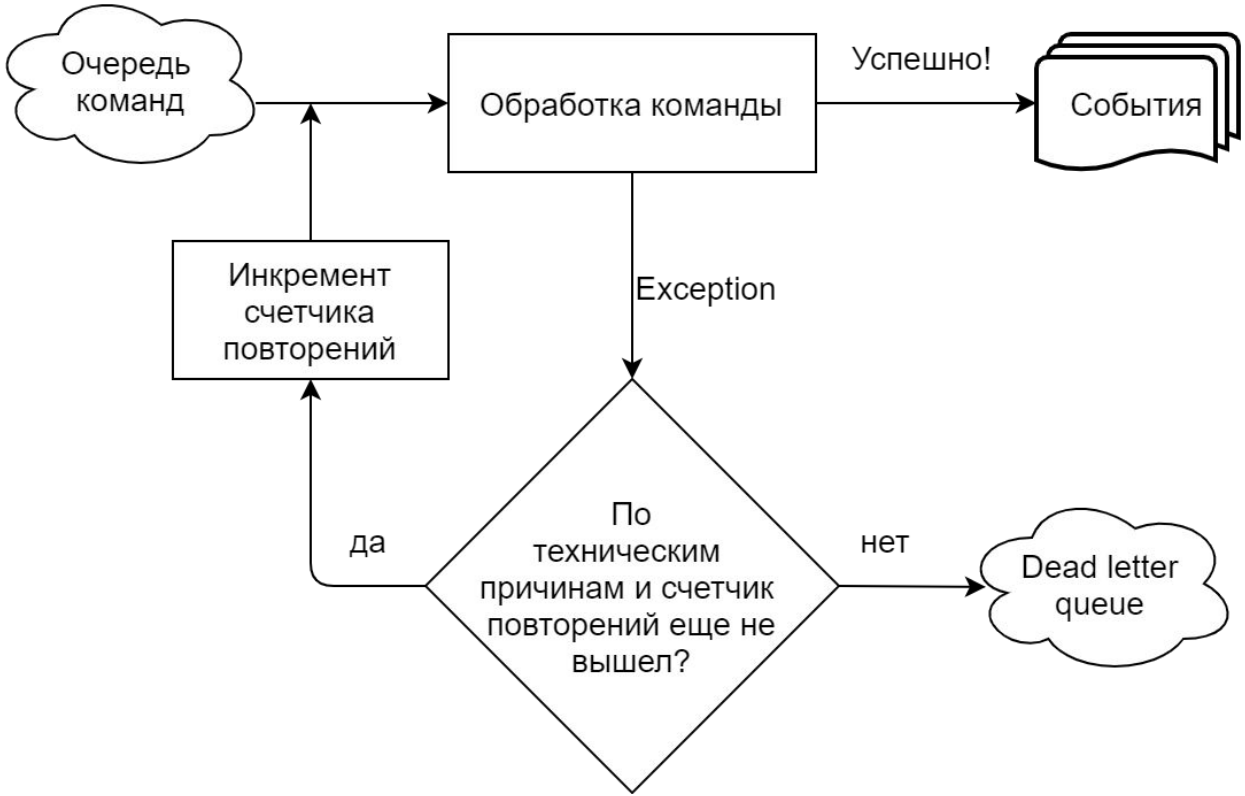
Команда 1

Команда 2

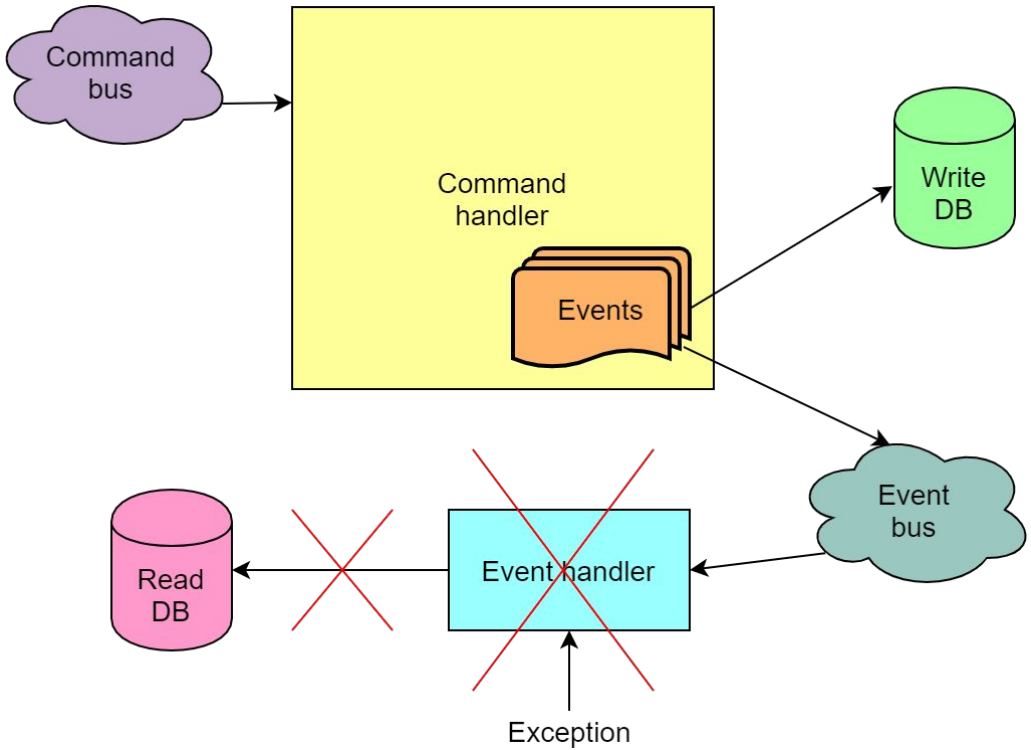


Агрегат 1

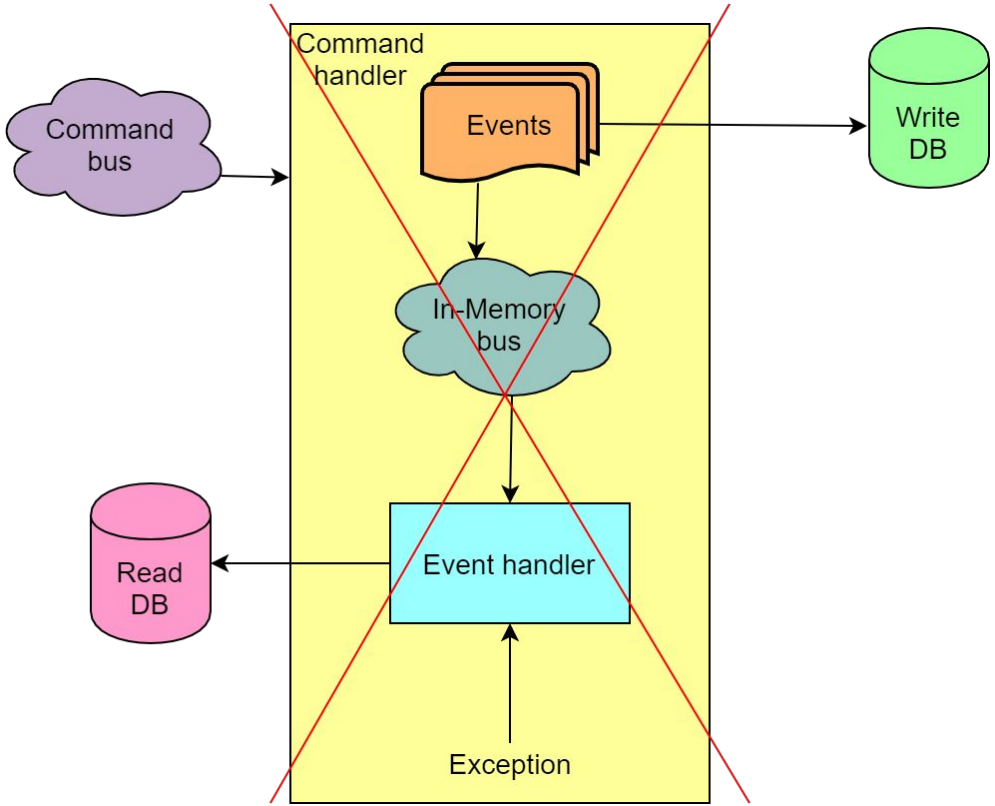
# Ошибки обработки команды из очереди



# Ошибки обработки **событий** из очереди



# Ошибки обработки событий из очереди. Костыль

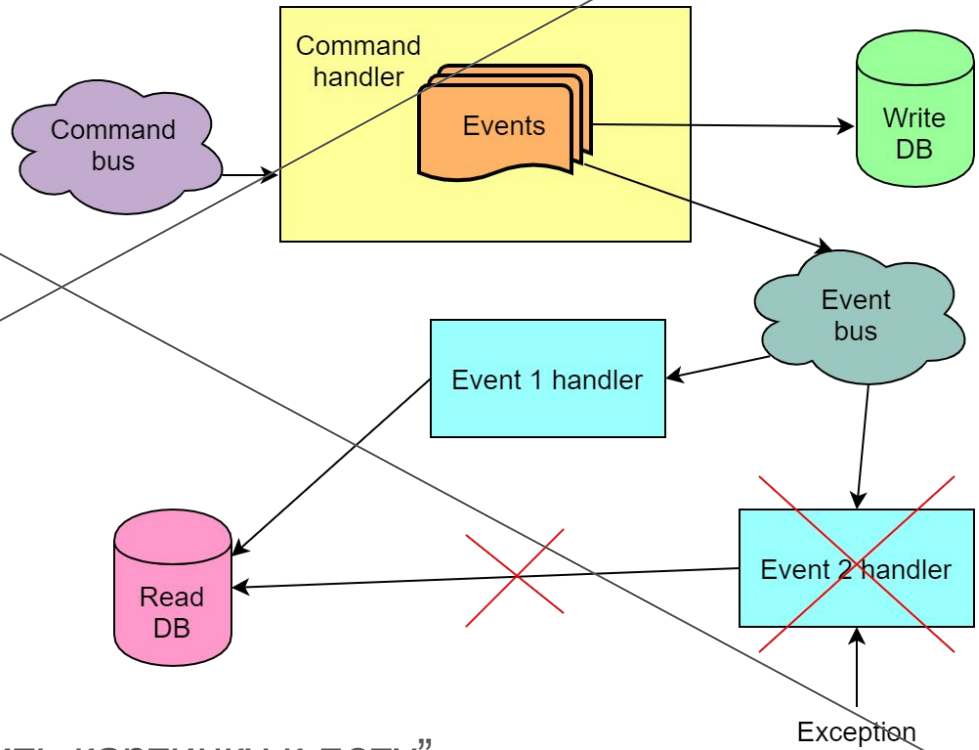


# Ошибки обработки событий из очереди. Выводы

- Вся бизнес-логика должна быть в обработчиках команд
- Обработчики событий не должны падать

# Параллельная обработка событий команды

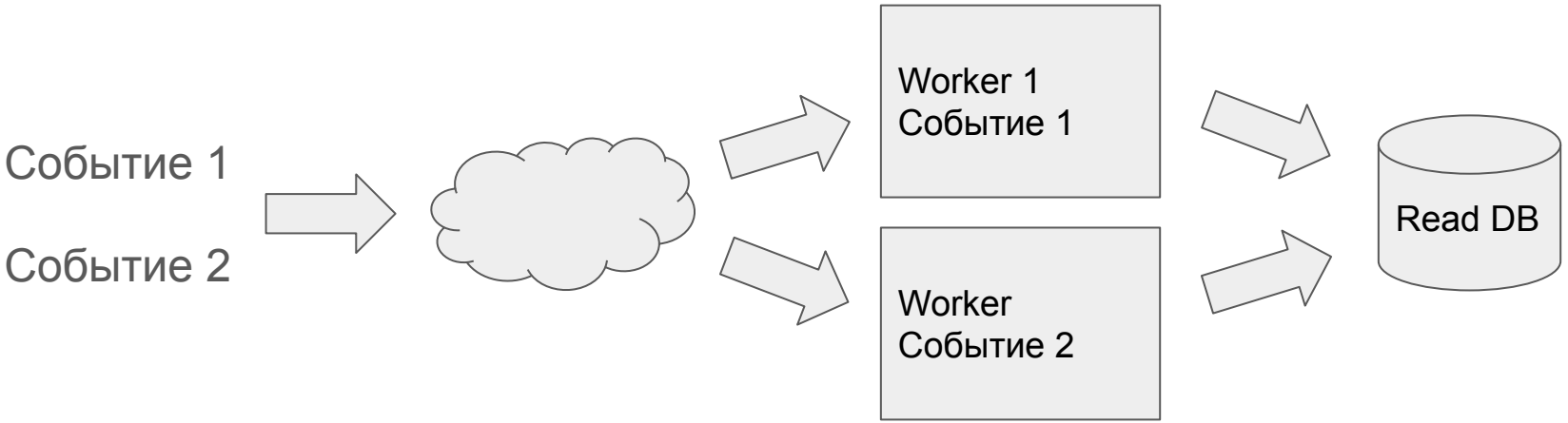
- Одна команда - много событий
- Несколько одновременных команд - очень много событий




Пример: “создать картинку” и “добавить картинку к лоту”

# Параллельная обработка событий команды

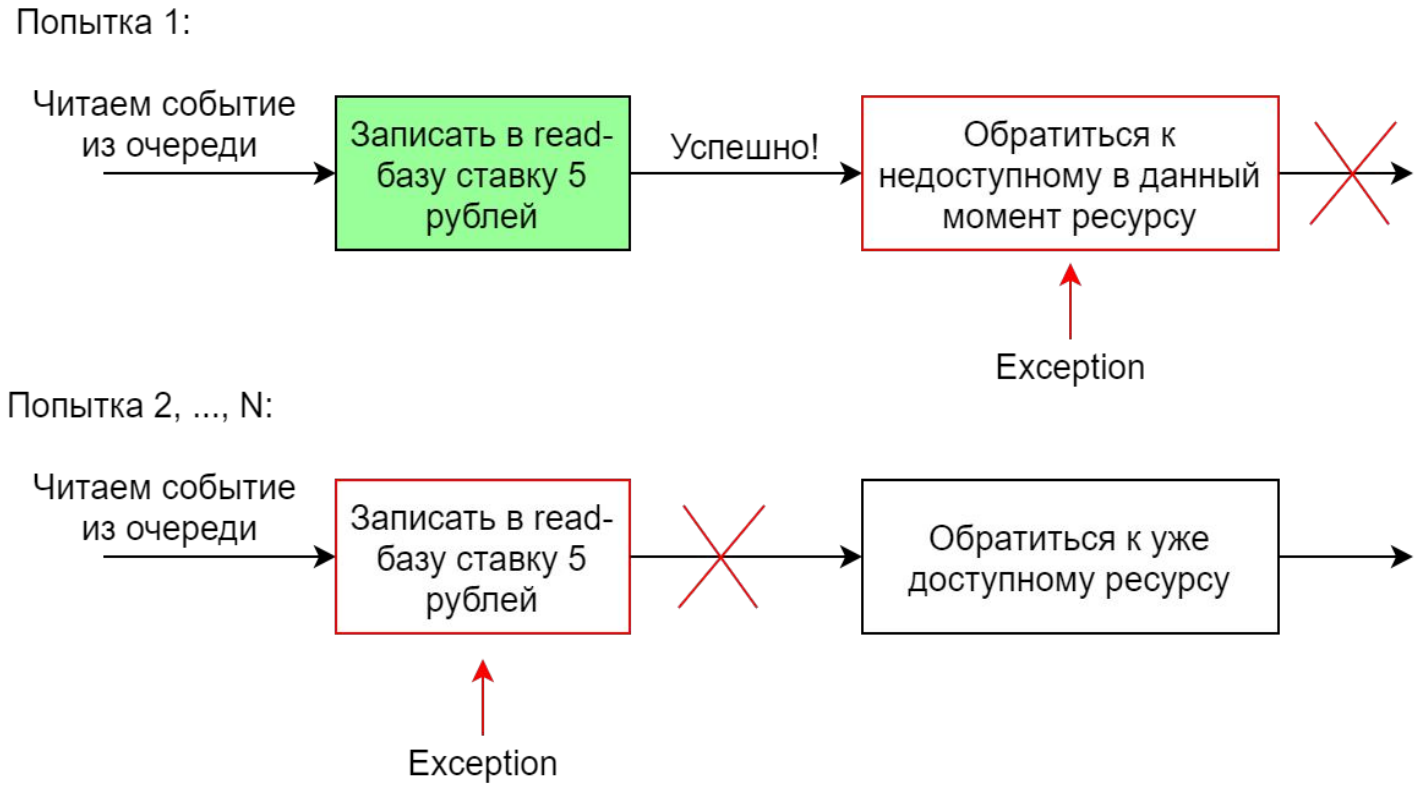
- Одна команда - много событий
- Команда “добавить картинку к лоту”
- События: “создать картинку” и “добавить картинку к лоту”



# Параллельная обработка событий команды. Выводы

- Последовательность событий важна
- Очередь с гарантией FIFO
- Пакетная отправка событий

# Обработка события несколькими обработчиками



# Обработчики событий должны быть идемпотентными

# Выводы

- Выбирать фреймворк без спешки (если вообще выбирать)
- Делить на агрегаты аккуратно
- Бизнес-логика - в обработчиках команд!
- Обработчики событий не должны падать
- Обработчики событий должны быть идемпотентными
- Использовать очередь с гарантией FIFO и наличием пакетного режима

# Ссылки

- https://habr.com/ru/company/arcadia/blog/509426/ - моя статья на Хабре
- https://habr.com/ru/post/146429/ - введение в CQRS и Event Sourcing, не моя статья

# Вопросы