СЛЁРМ Образование для инженеров и технических лидеров

Проектирование микросервисов на примере разделения монолита

Спикер: Щербаков Пётр
Solution Architect

План урока

01 Синхронные и асинхронные

02 Классические и событийные

03 Интеграция через посредника

Синхронные и асинхронные

Синхронные и асинхронные

- HTTP(S) (REST) OpenAPI →
- SOAP XML XSD
- gRPC (Framework) HTTP/2.0 ProtoBuf Contract Codegen
- MQ (Kafka, Rabbit, etc.)
- Custom protocol over TCP

Синхронные и асинхронные

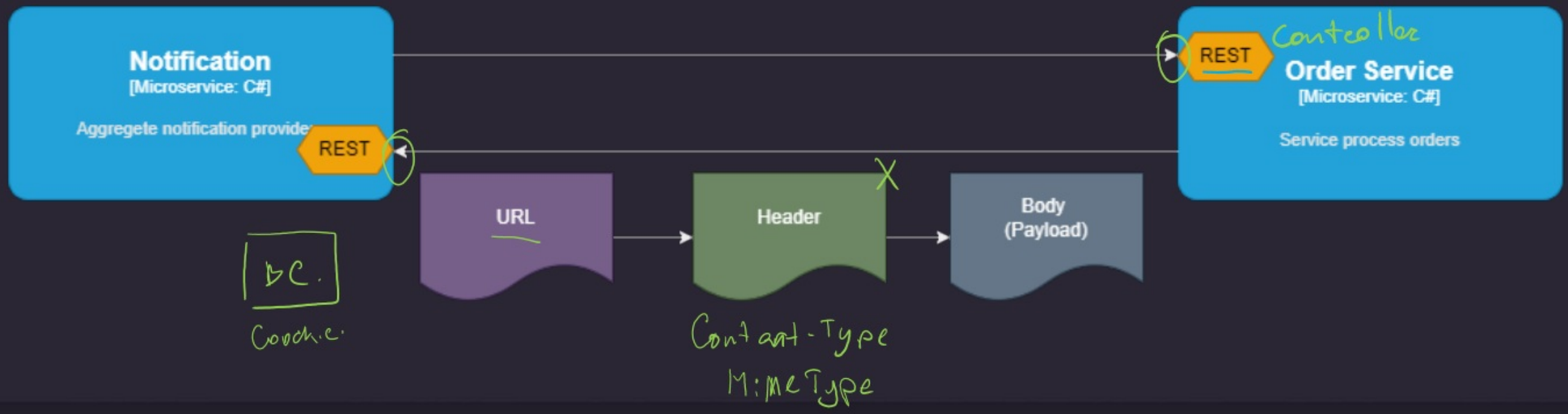
HTTP(S)/JSON -> REST, для веб-API(сервисов) - RESTful

1. Модель клиент-сервер – отделяем интерфейс от логики
2. Отсутствие состояния – вся информация в запросе
3. Кэширование – кэш ответов от сервера
4. Единообразие интерфейса – описание данных, форматов и метаданных
5. Сокрытие слоёв – не знаем с каким узлом работаем
6. Код по требованию – апплеты, скрипты грузим клиенту

Синхронные и асинхронные

HTTP(S)/JSON -> REST, для веб-API(сервисов) - RESTful

1. Модель клиент-сервер – отделяем интерфейс от логики
2. Отсутствие состояния – вся информация в запросе
3. Кэширование – кэш ответов от сервера
4. Единообразие интерфейса – описание данных, форматов и метаданных
5. Сокрытие слоёв – не знаем с каким узлом работаем
6. Код по требованию – апплеты, скрипты грузим клиенту



Синхронные и асинхронные

HTTP(S) -> REST, для веб-API(сервисов) - RESTful

1. Модель клиент-сервер
2. Отсутствие состояния
3. Кэширование
4. Единообразие интерфейса
5. Сокрытие слоёв
6. Код по требованию

Coes

Методы
- Options
- Get
- Post
- Put
- Patch
- Delete
- Connect
- Trace

URL
- Схема
- Логин
- Пароль
- Хост
- Порт
- Путь
- Запрос
- Якорь

Headers
- Более 50
- Список можно расширить
- Делятся на заголовки запросов, ответов и сущности

Синхронные и асинхронные

Методы
- Options
- Get
- Post
- Put
- Patch
- Delete
- Connect
- Trace

URL
- Схема
- Логин
- Пароль
- Хост
- Порт
- Путь
- Запрос
- Якорь

https://login:password@host:port/api/v1/path?query=param#anchor

Headers
- Content-Type
- Host
- Location
- User-Agent

GET https://some.example.com/api/v1/orders/{clientId}

GET https://some.example.com/api/v1 /{clientId}/orders

GET https://some.example.com/api/v1/orders?client={clientId}

POST https://some.example.com/api/v1/orders

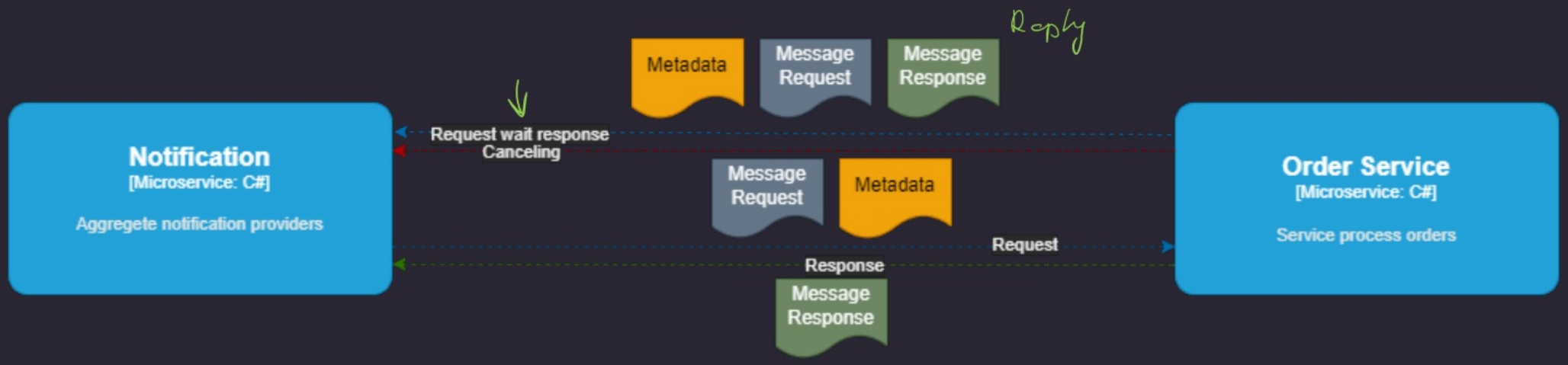
Payload

Header

Синхронные и асинхронные

gRPC/protobuf

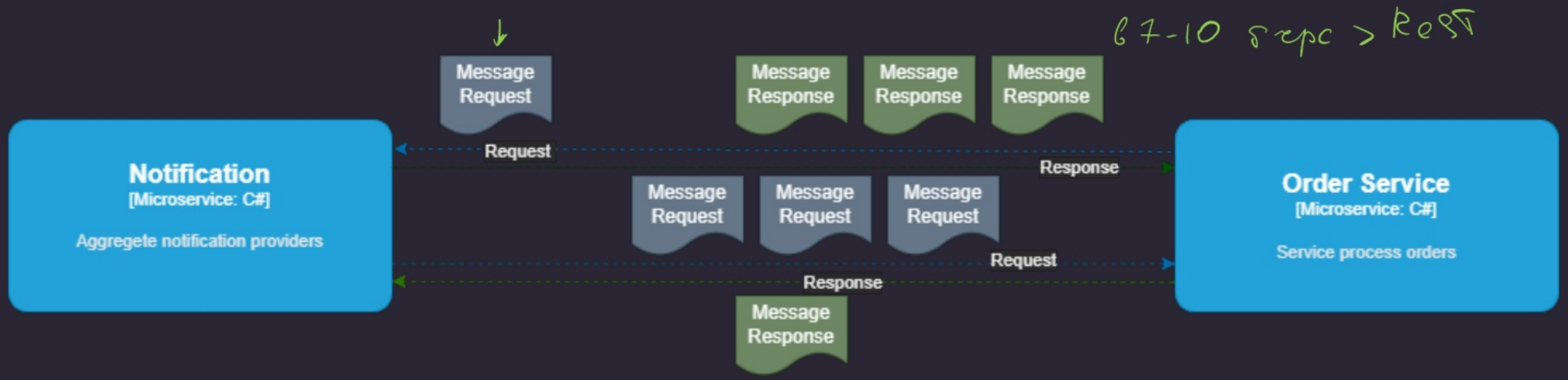
1. Модель клиент-сервер – отделяем интерфейс от логики
2. Синхронный и асинхронный режим работы
3. Описание запроса в метаданных
4. Дедлайны, таймауты, отмена запроса
5. Поддерживает режим стриминга
6. Может работать в двунаправленном режиме



Синхронные и асинхронные

gRPC/protobuf - идеология

- Пишем код сервиса (контракт)
- Контракт состоит из методов, сообщений запросов и ответов
- Вызываем protoc, генерируемм код на нужном языке
- Описываем логику
- Запускаем
- Интегрируемся



Синхронные и асинхронные

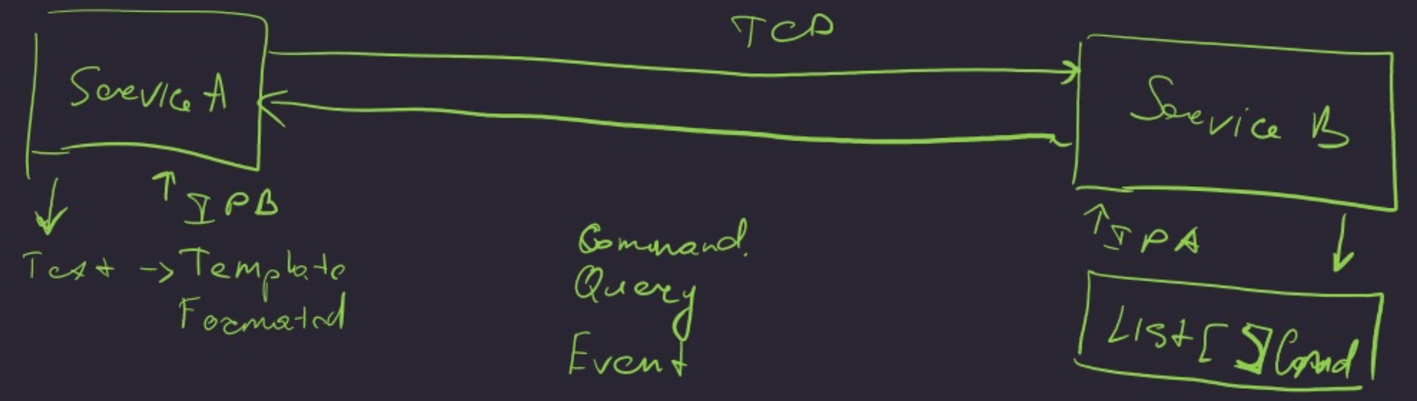
Kafka/RabbitMQ

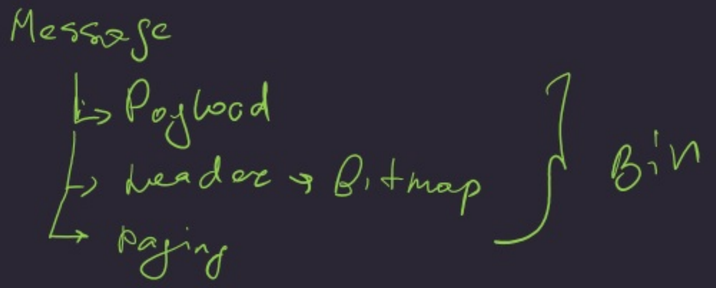
- Модель клиент-посредник-сервис
- Контракт состоит из команд и событий
- Асинхронный режим взаимодействия

Синхронные и асинхронные

Custom protocol over TCP

- Модель клиент-сервис
- Мы само создаем контракт, придумываем сущности для обмена
- Решаем каким будет режим взаимодействия





Классические и событийные

Классические и событийные

- Request-Response → 1
- Webhook/Callback → Request → Request
  ↳ POST ↳ call
- Event-Driven Architecture
  Saga Pattern

Классические и событийные

Request - Response

1. Модель клиент-сервер
2. Синхронный режим работы
3. Есть таймауты

Классические и событийные

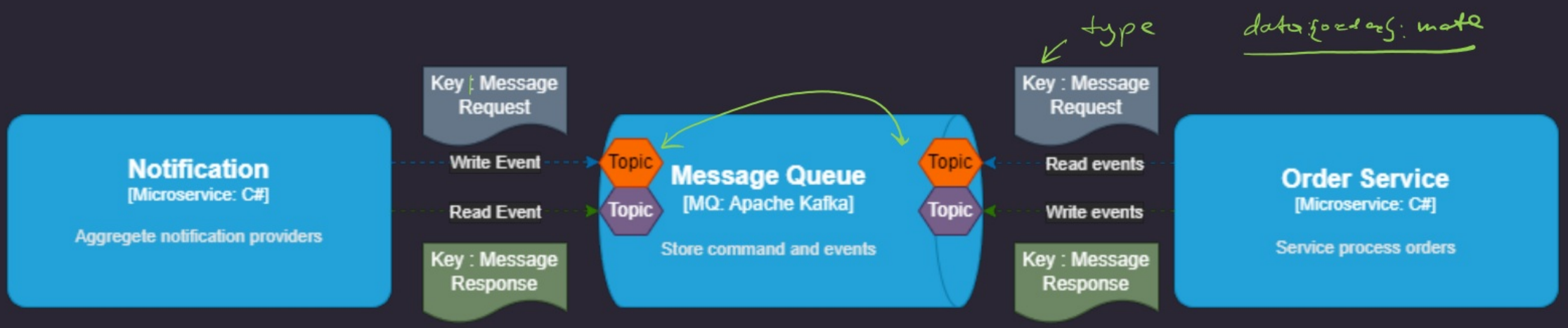
Webhook/Callback

1. Модель клиент-сервер
2. Асинхронный режим работы
3. Есть таймауты
4. Необходим адрес ответа

Классические и событийные

Event-Driven

1. Модель клиент-черный ящик
2. Асинхронный режим работы
3. Событийная модель обмена данными
4. Необходимо определить очереди ответа и запросов



Классические и событийные

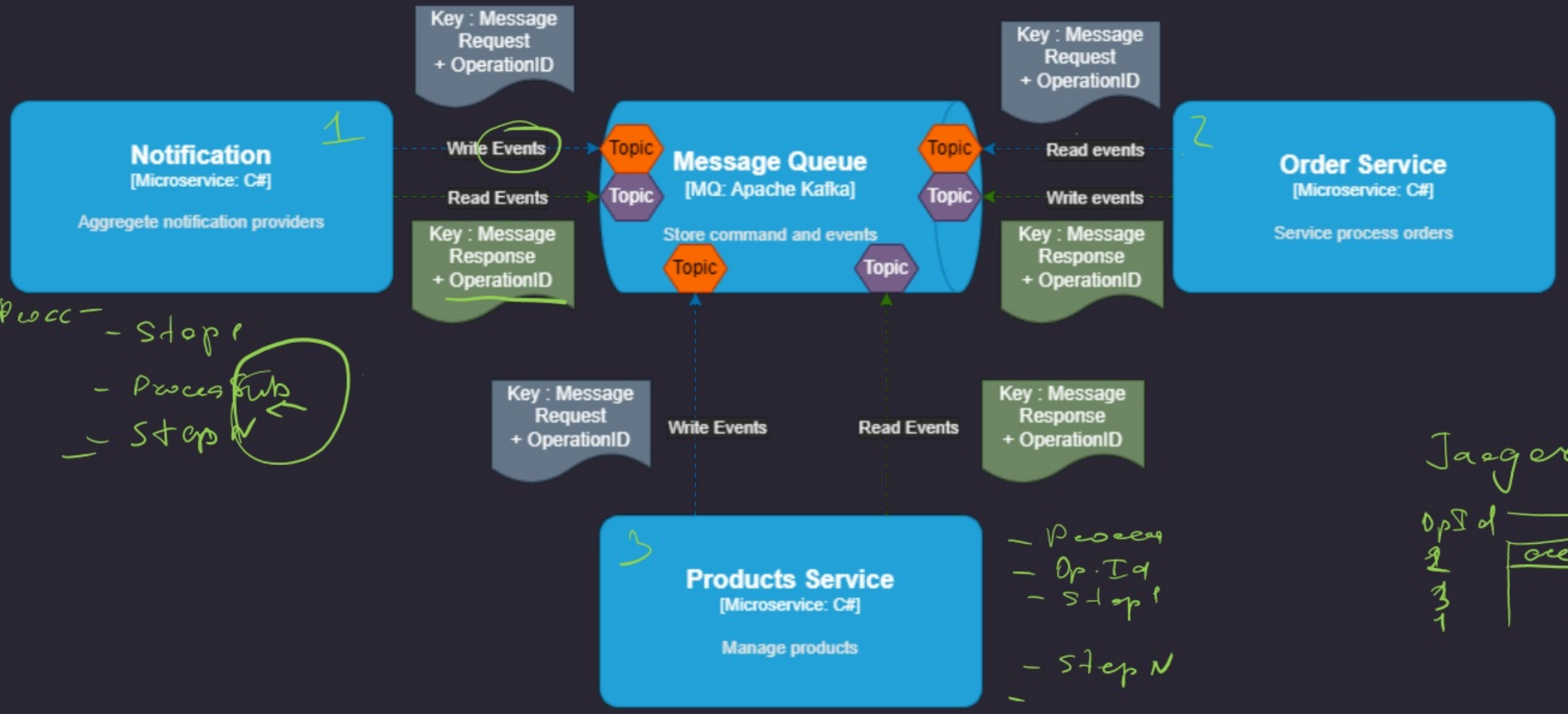
SAGA - Pattern

Distributed Transaction

Orchestration

Choreography

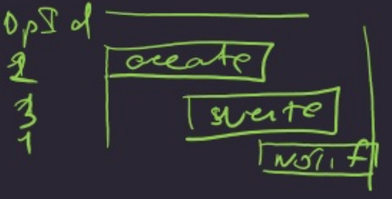
BPMS



Local Proc
- Step
- Process Sub
- Step

- Process
- Op. Id
- Step 1
- Step N

Jaeger Tracing



СЛЁРМ

Интеграция через посредника

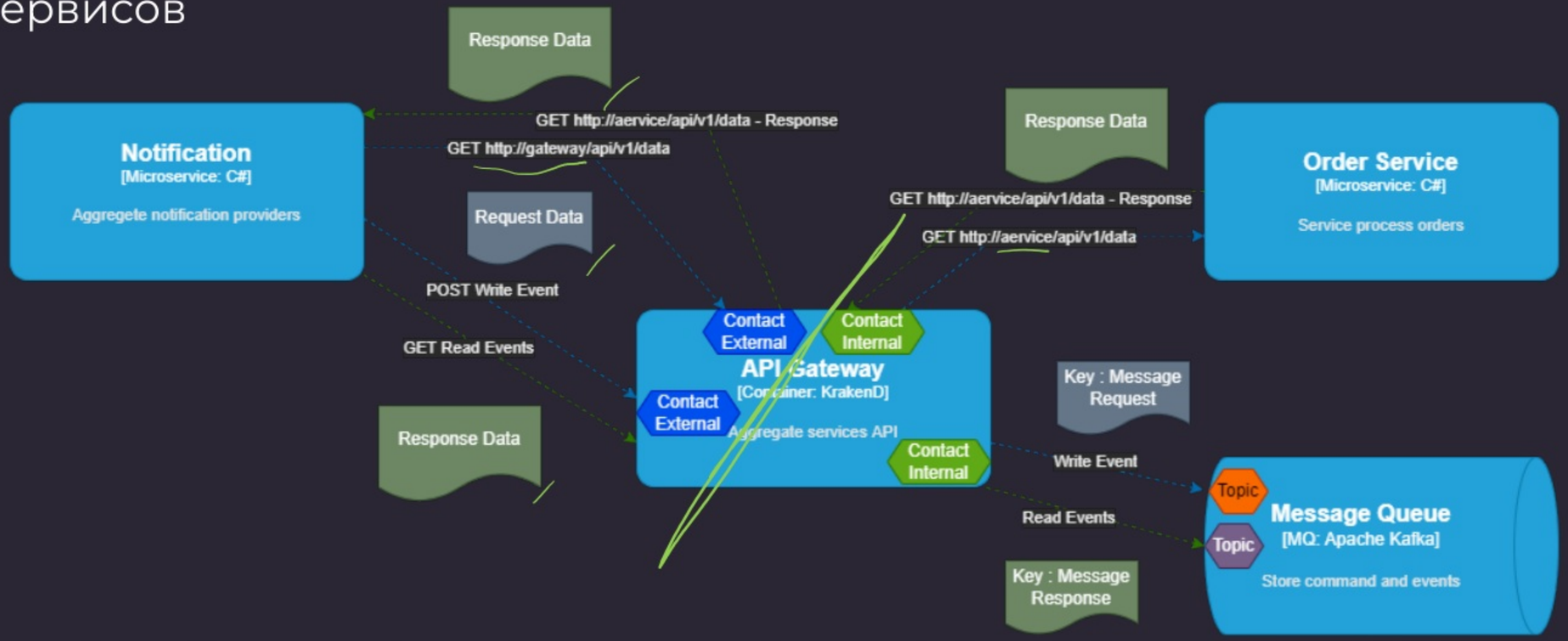
Интеграция через посредника

- API Gateway
- GraphQL
- Database/S3

Интеграция через посредника

API Gateway

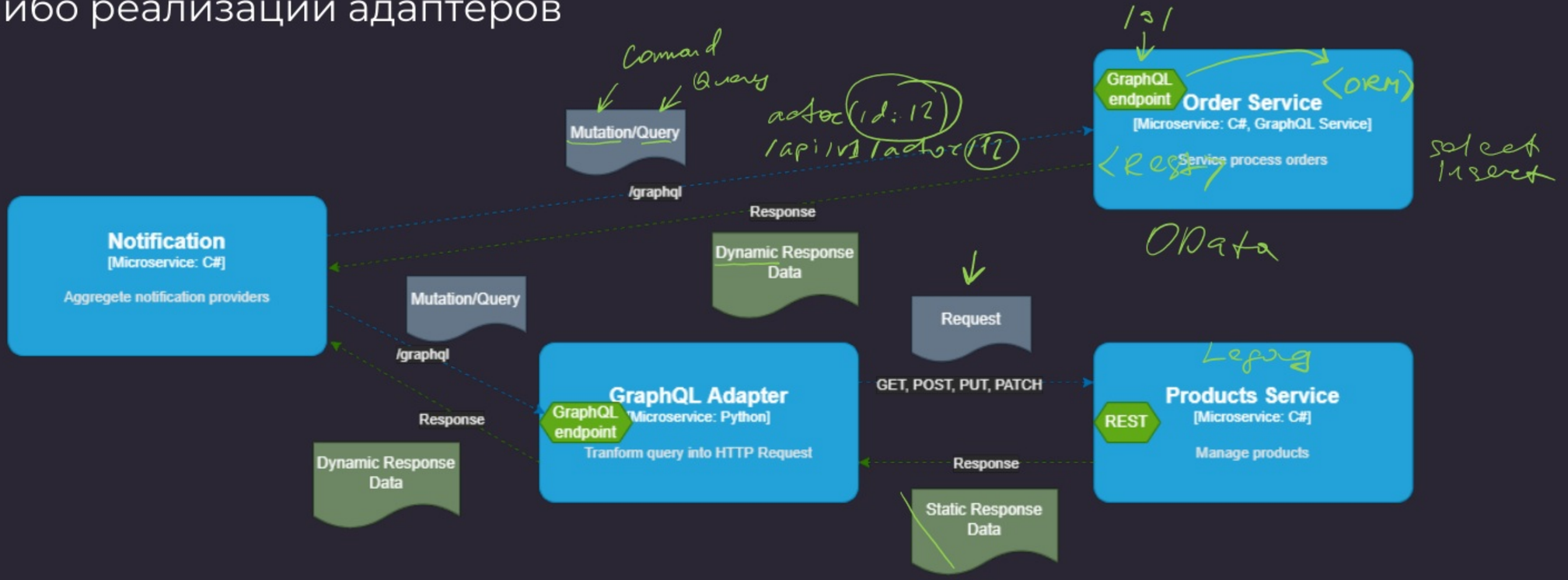
1. Модель клиент-посредник-сервер
2. Синхронный и асинхронный режимы работы
3. Привязан к модели данных
4. Точное определение точек доступа
5. Необходимо описать маршрутизацию и данные, не требует изменения сервисов



Интеграция через посредника

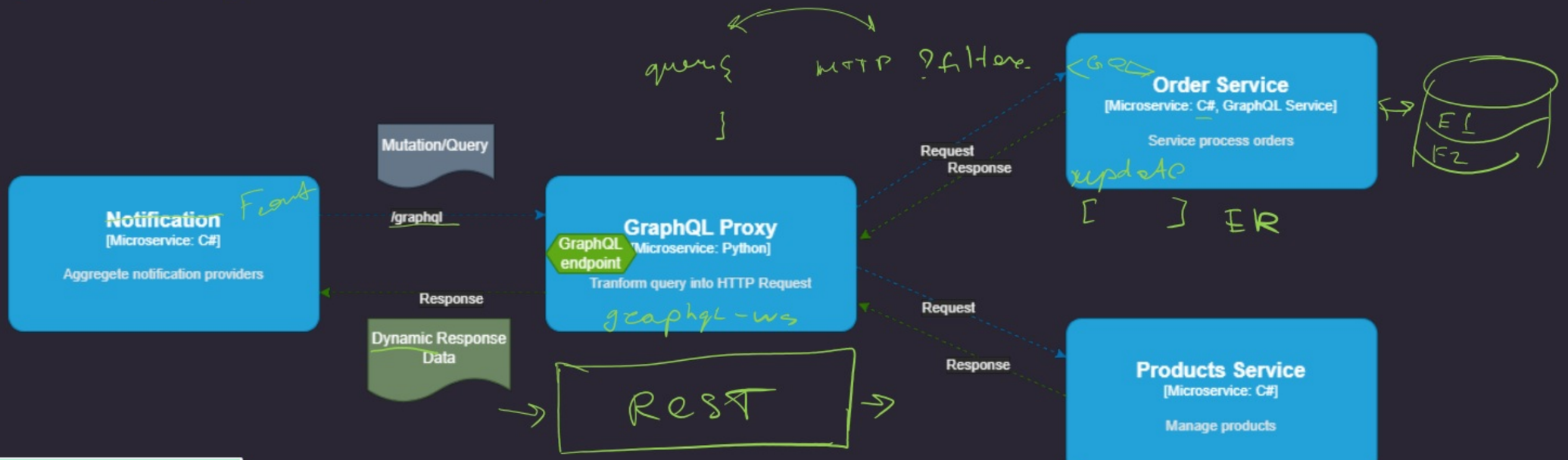
GraphQL

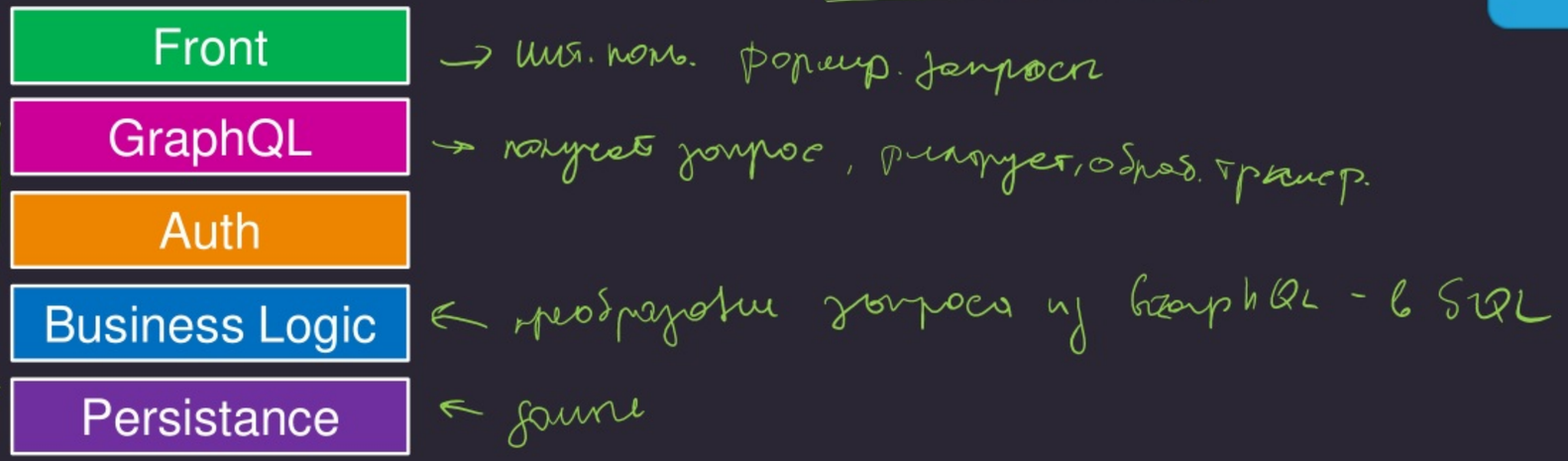
1. Модель клиент-посредник-сервер
2. Синхронный режимы работы
3. Привязан к модели данных
4. Не точное определение точек доступа
5. Необходимо описать маршрутизацию и данные, требует изменения сервисов, либо реализации адаптеров



Интеграция через посредника

GraphQL как централизованное решение

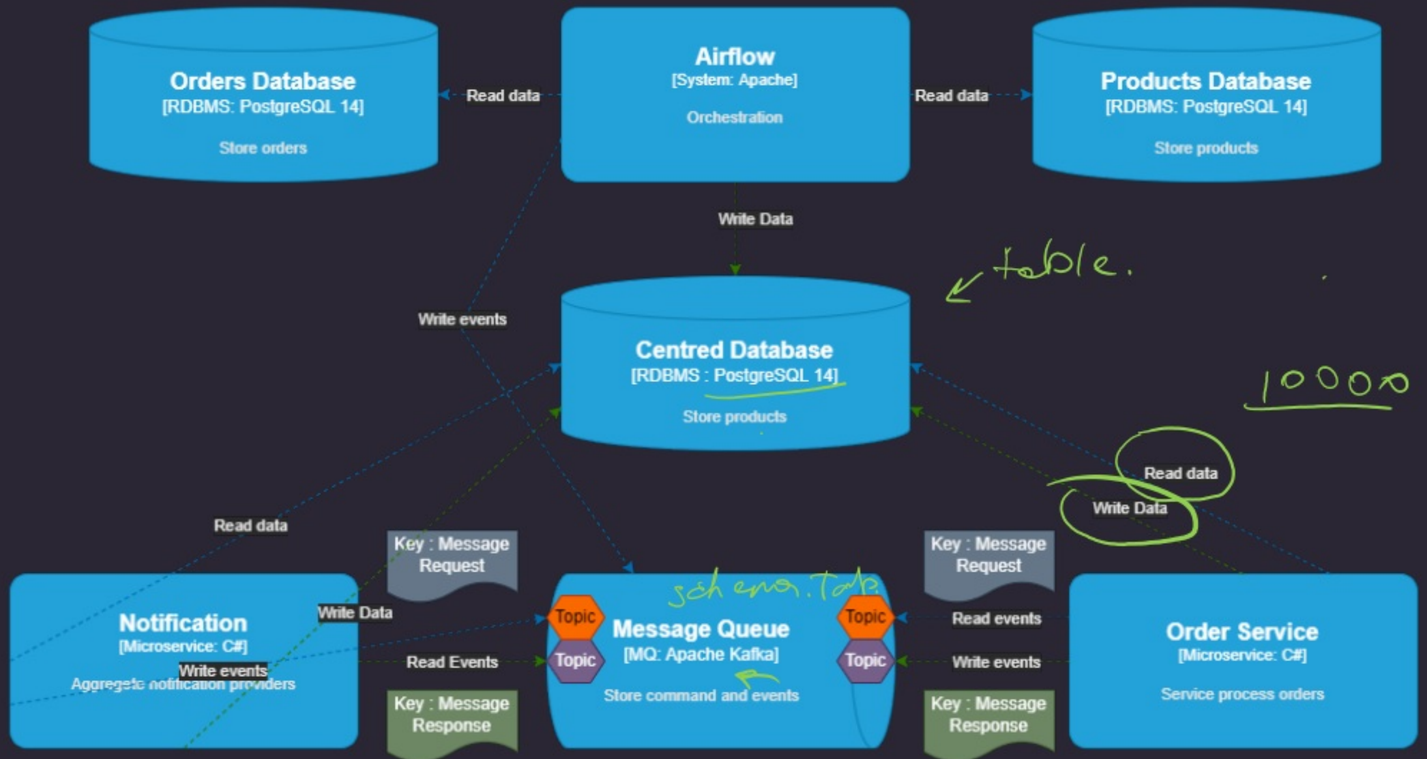


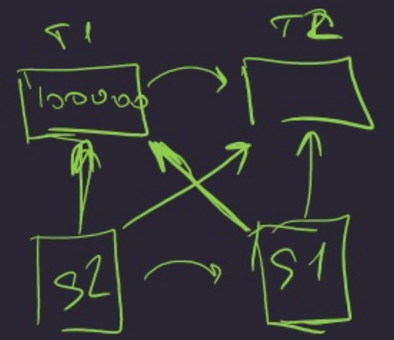


Интеграция через посредника

Database/S3

1. Модель клиент-посредник-сервер
2. Асинхронный режимы работы
3. Привязан к модели данных
4. Явная точка доступа
5. Необходимо фиксировать формат данных и уведомления об их обновлении





The END;