

Куча страшных зверей



Окулов Антон

R.class

RabbitMQ



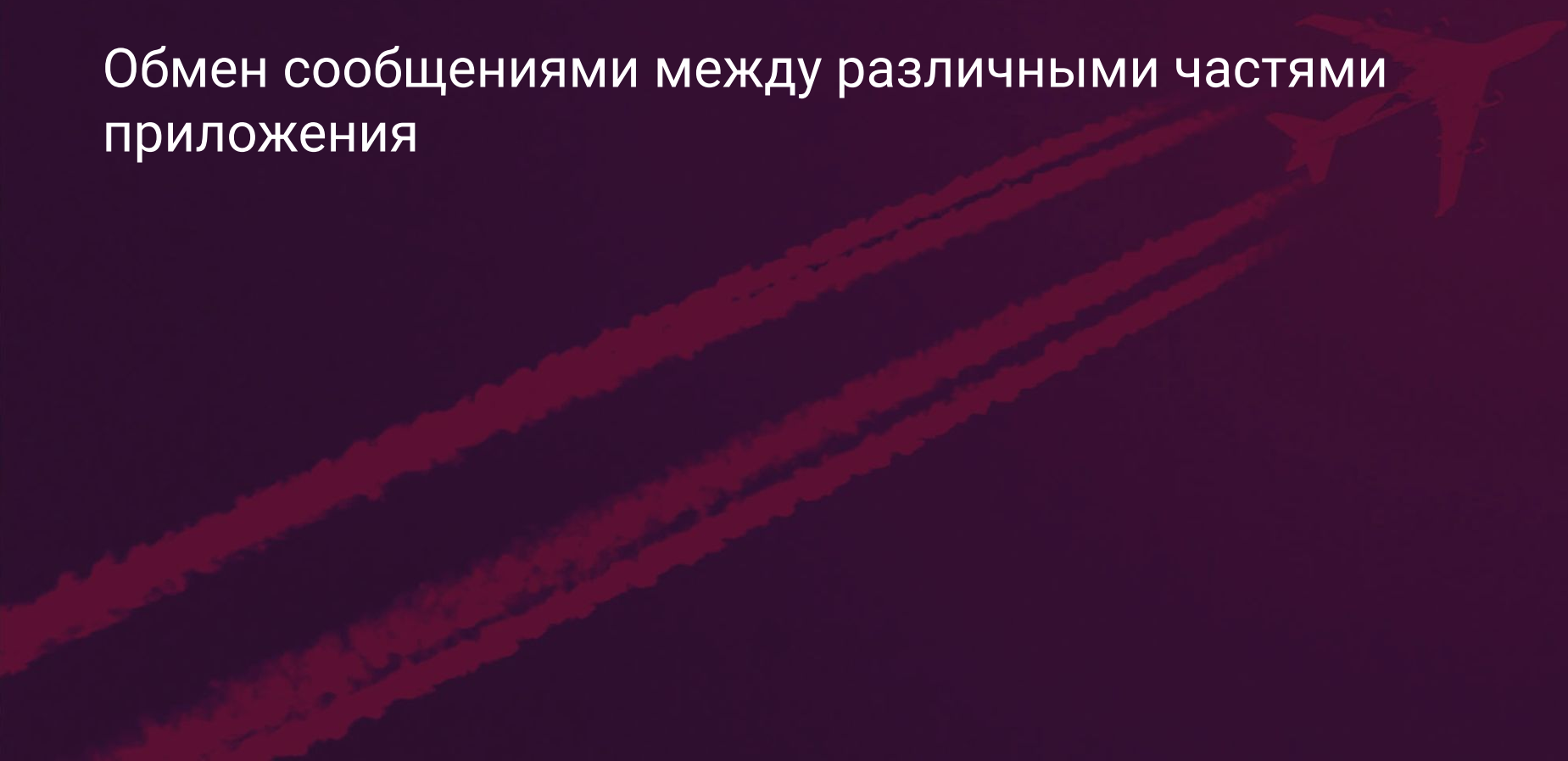
Что за зверь:

программный брокер сообщений на основе стандарта AMQP

Язык разработки: Erlang

Основная задача

Обмен сообщениями между различными частями приложения



Схематично



Схематично



Схематично



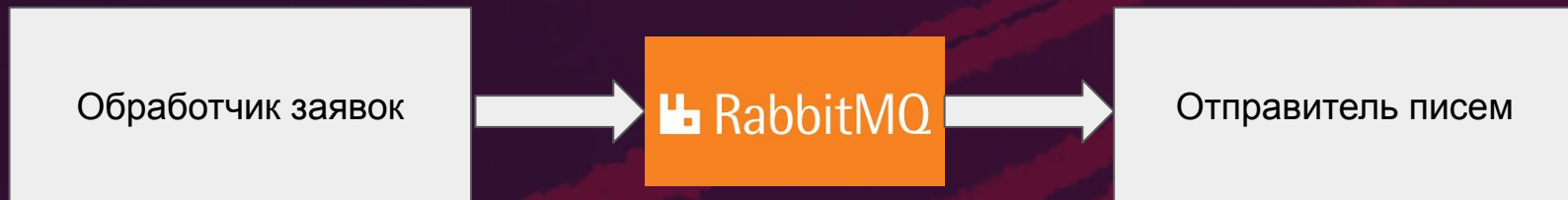
Схематично



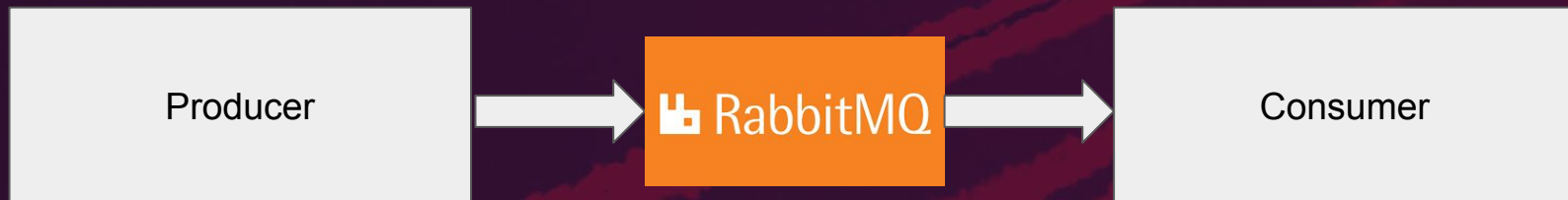
Схематично



Схематично



Схематично



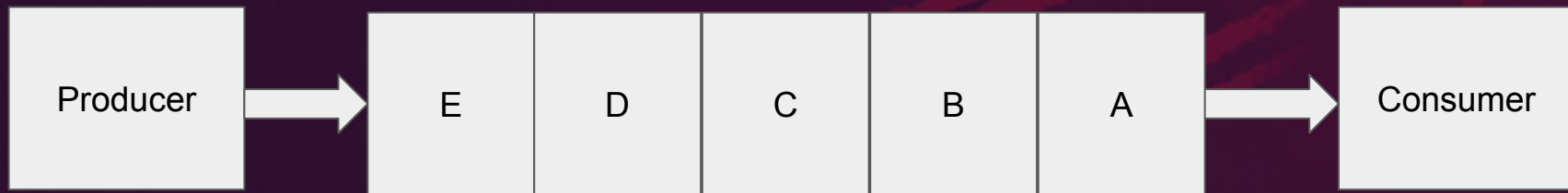
Схематично

Producer отправляет сообщения в RabbitMQ.

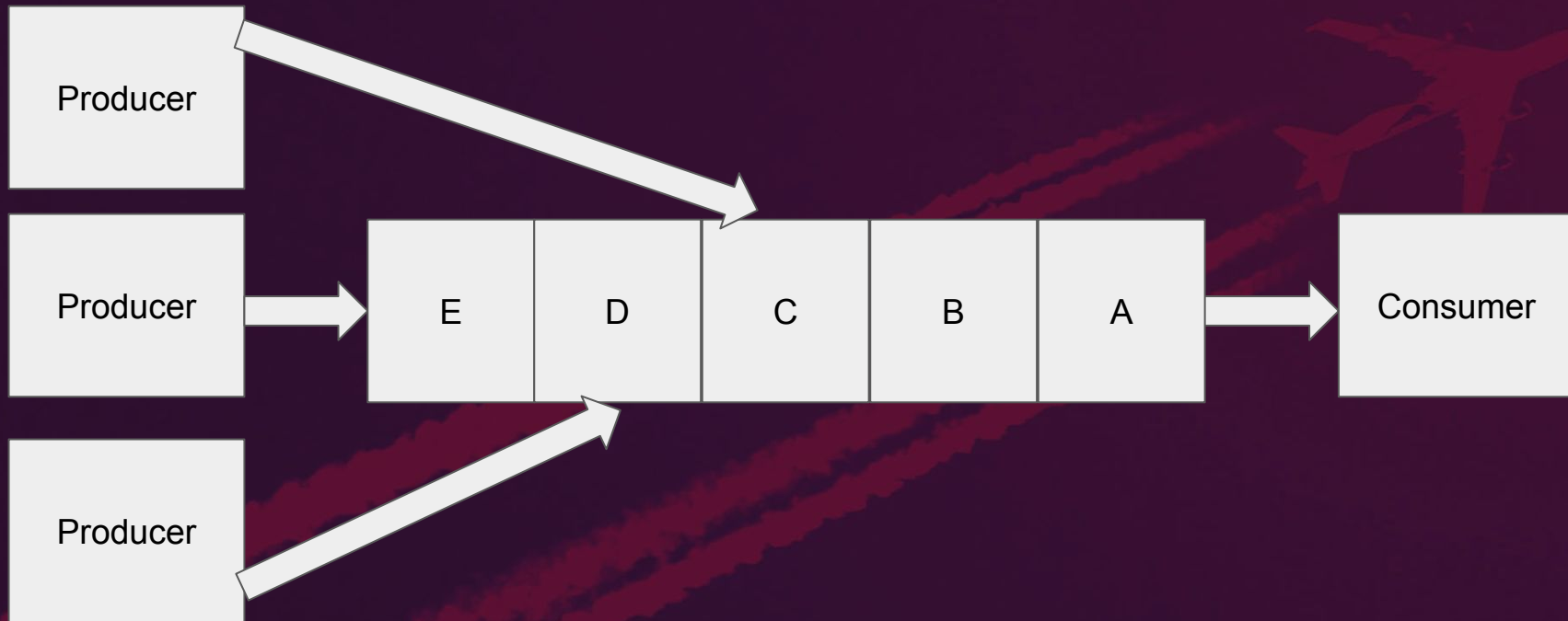
Consumer подключается к RabbitMQ и ожидает сообщений.



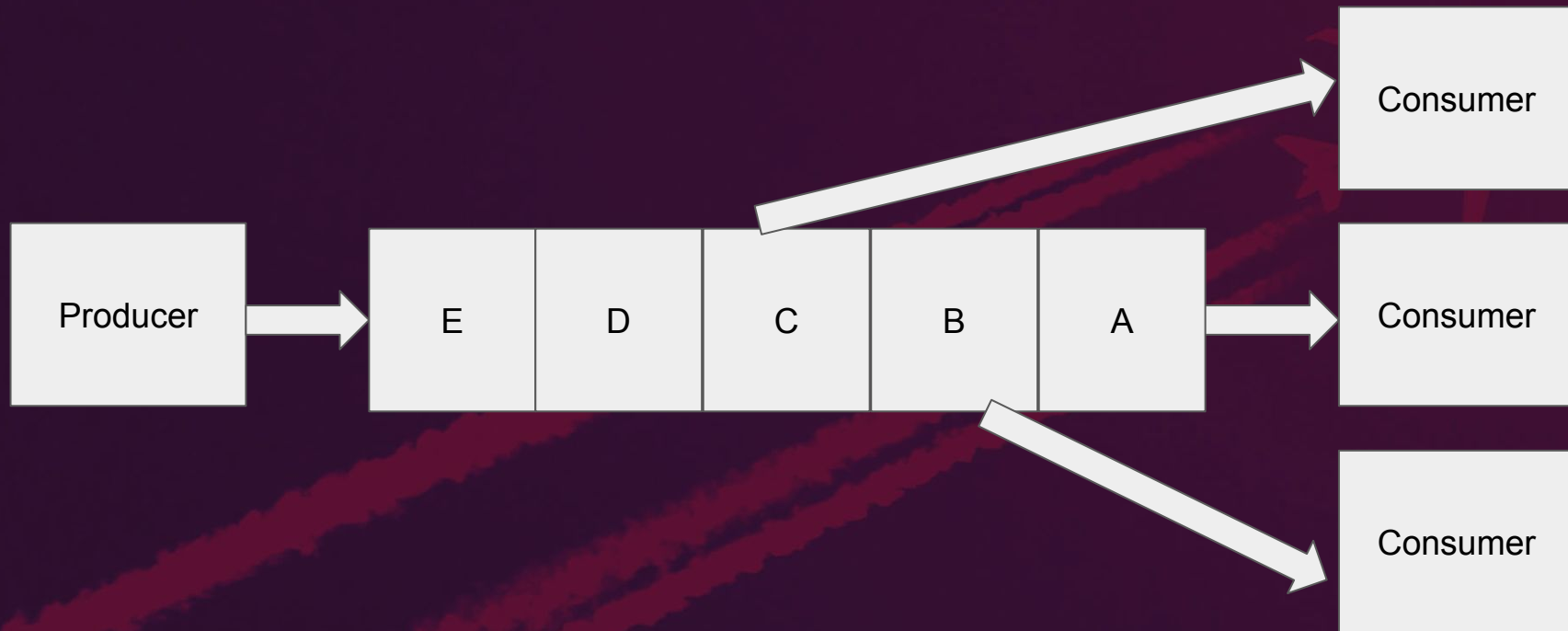
Схематично



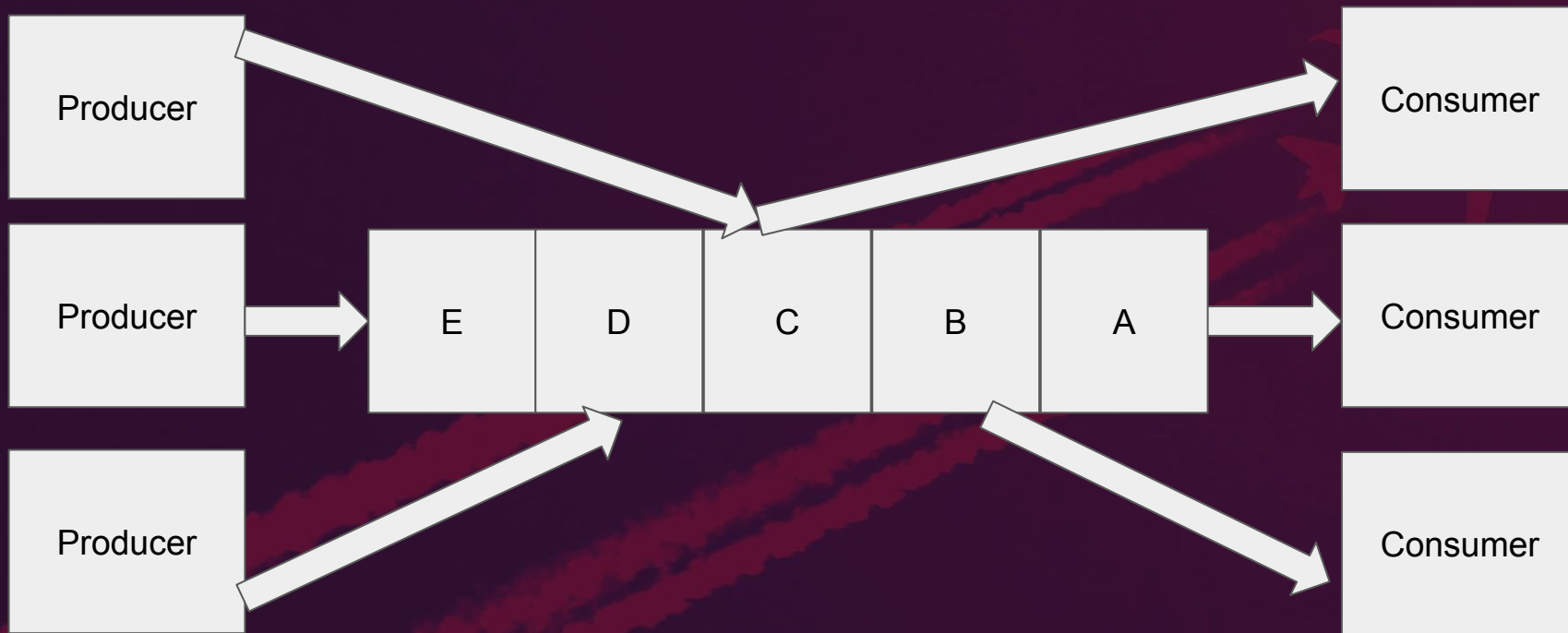
Схематично



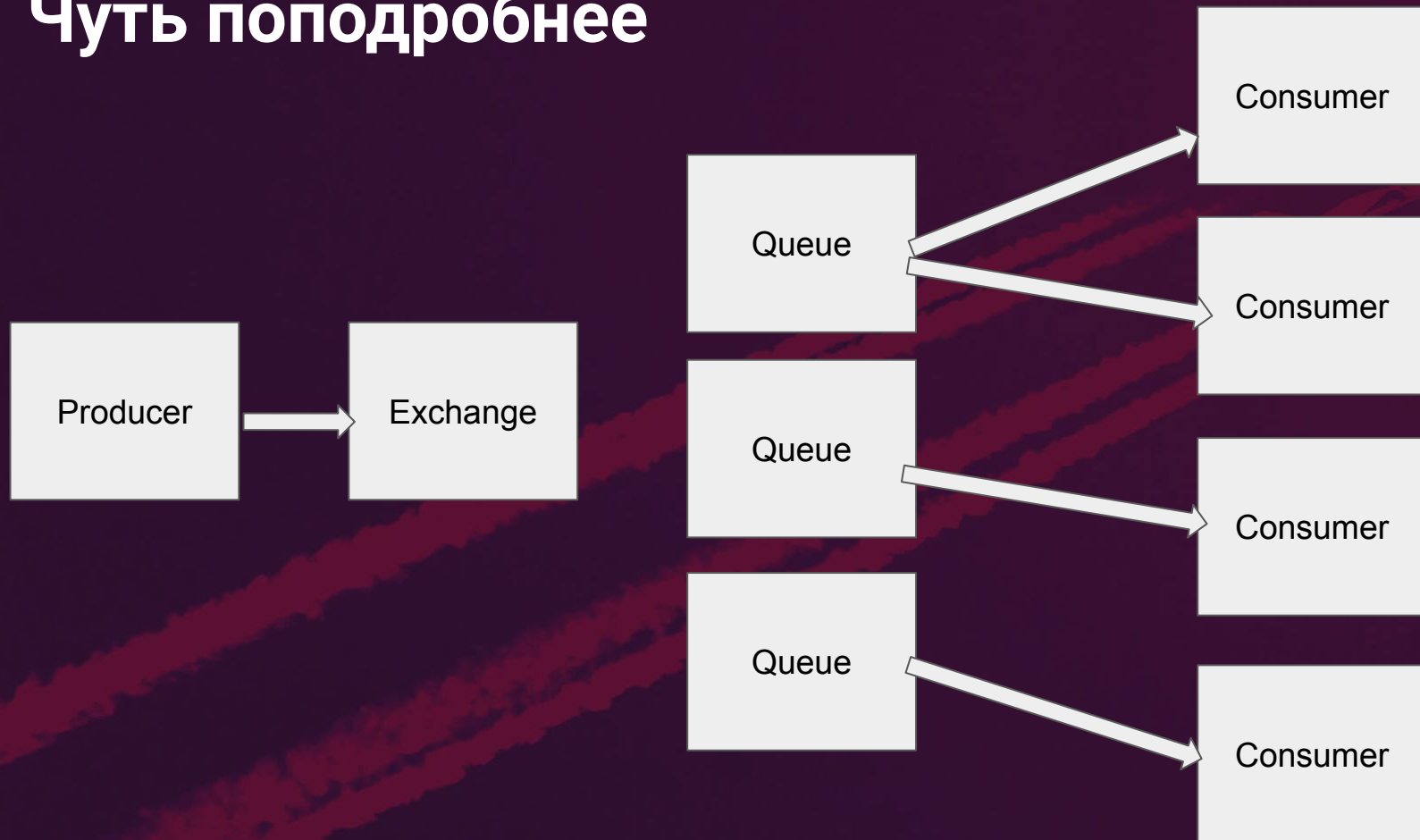
Схематично



Схематично



Чуть поподробнее



Чуть поподробнее

Message
text: ?
routing_key: log.error

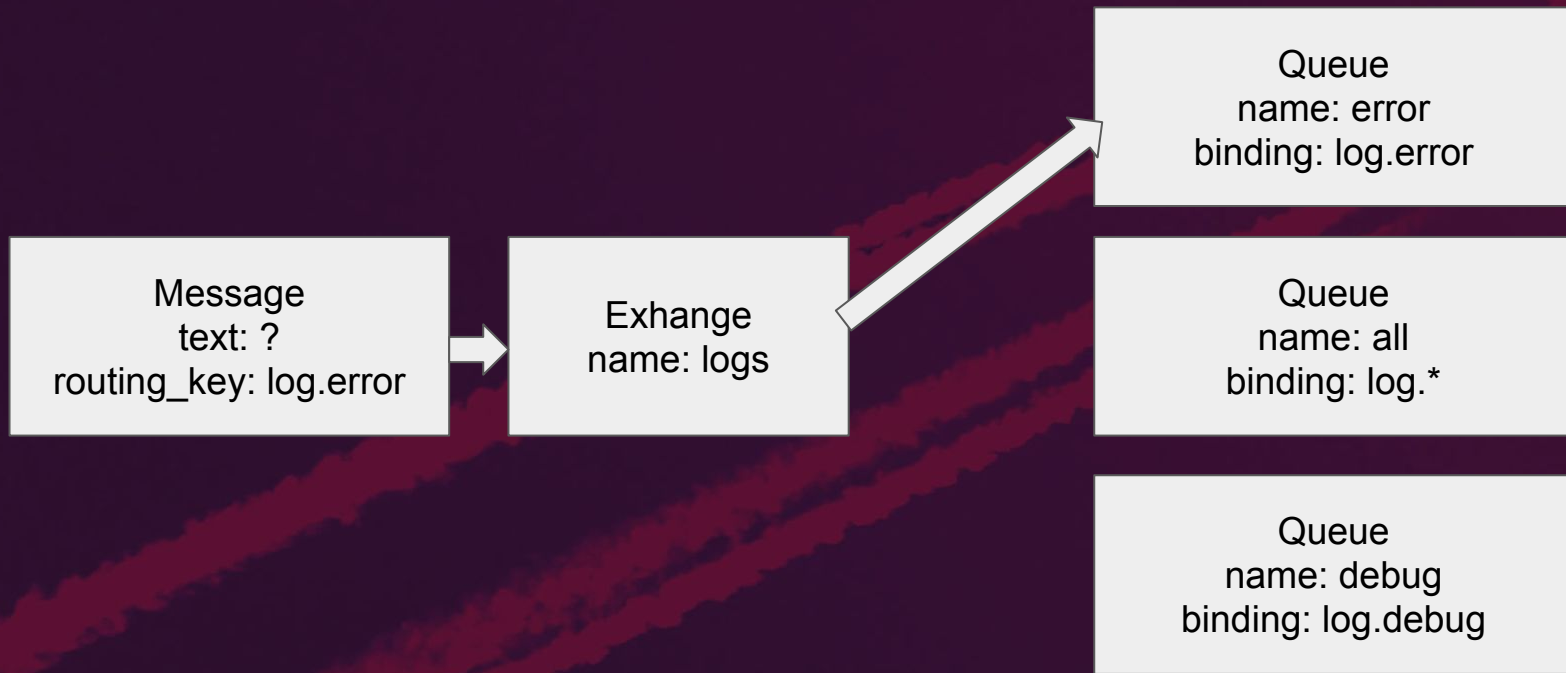
Exchange
name: logs

Queue
name: error
binding: log.error

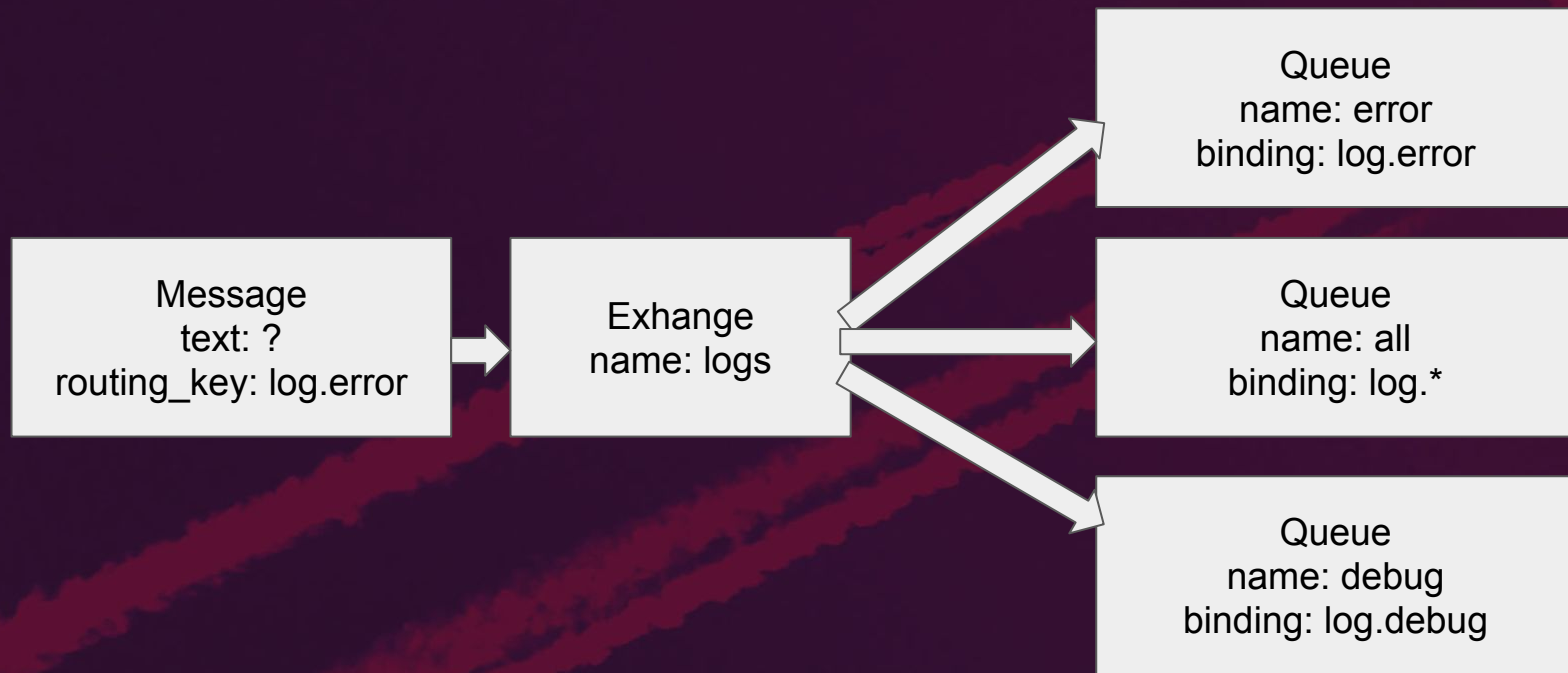
Queue
name: all
binding: log.*

Queue
name: debug
binding: log.debug

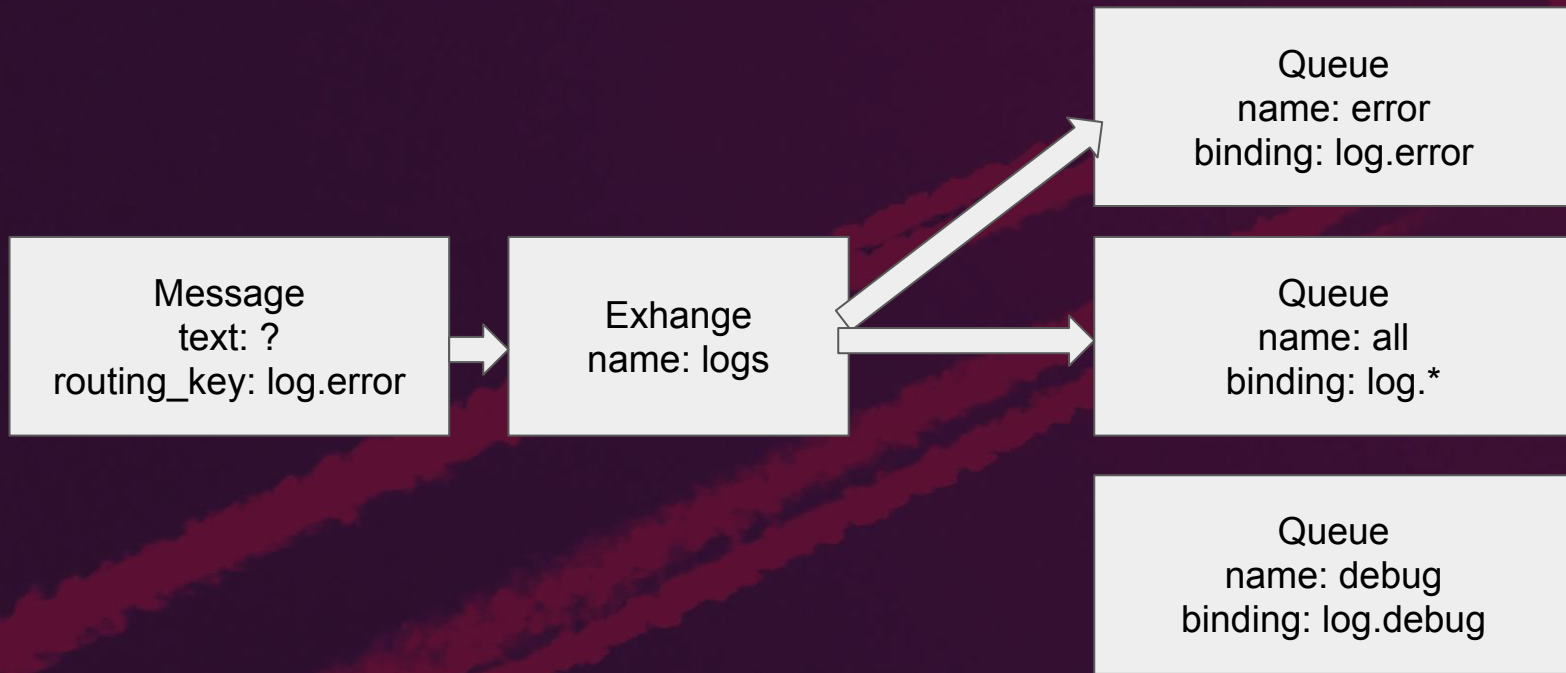
Direct Exchange Type



Funout Exchange Type



Topic Exchange Type



Фи́чи

- Durable Queue / Durable Exchange / Persistent message
 - сообщения, которые мы не хотим потерять при перезагрузке
- Ack - подтверждение обработки
- Dead Letters
- ...

Apache Kafka



Что за зверь:

распределённый программный брокер
сообщений

Язык разработки: Scala и Java

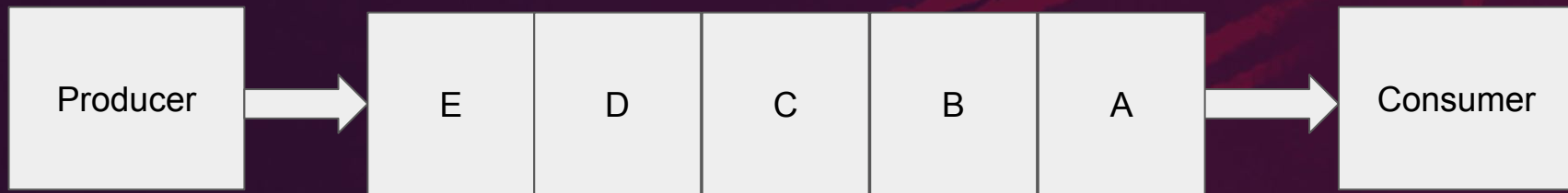
Основная задача

Обмен сообщениями между различными частями приложения

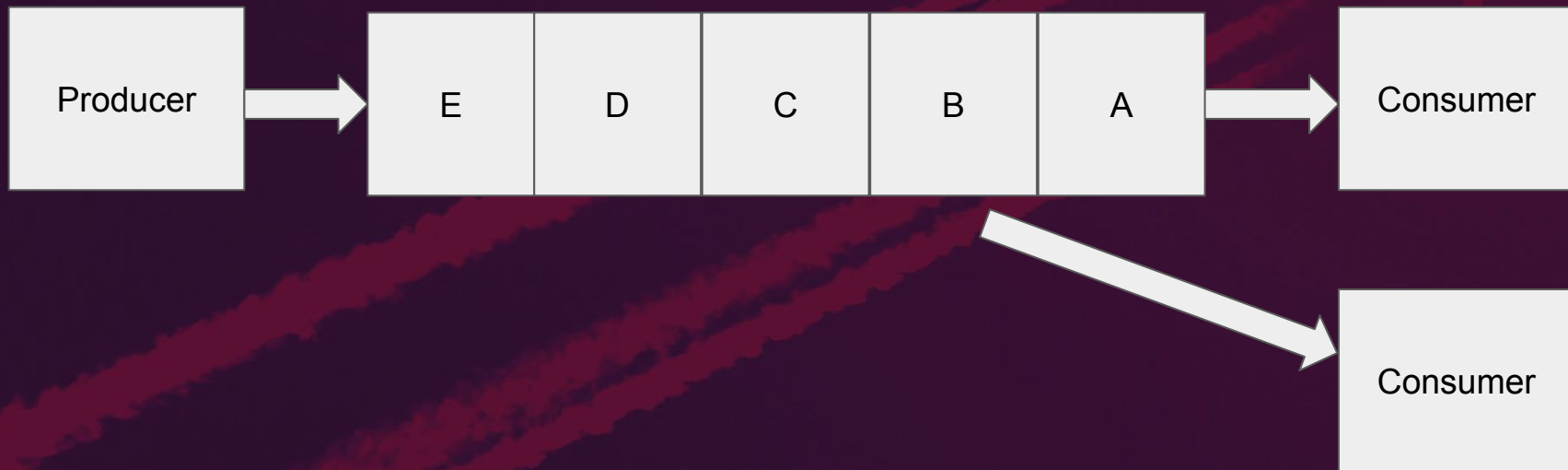
Одной из особенностей реализации инструмента является применение техники, сходной с журналами транзакций, используемыми в системах управления базами данных



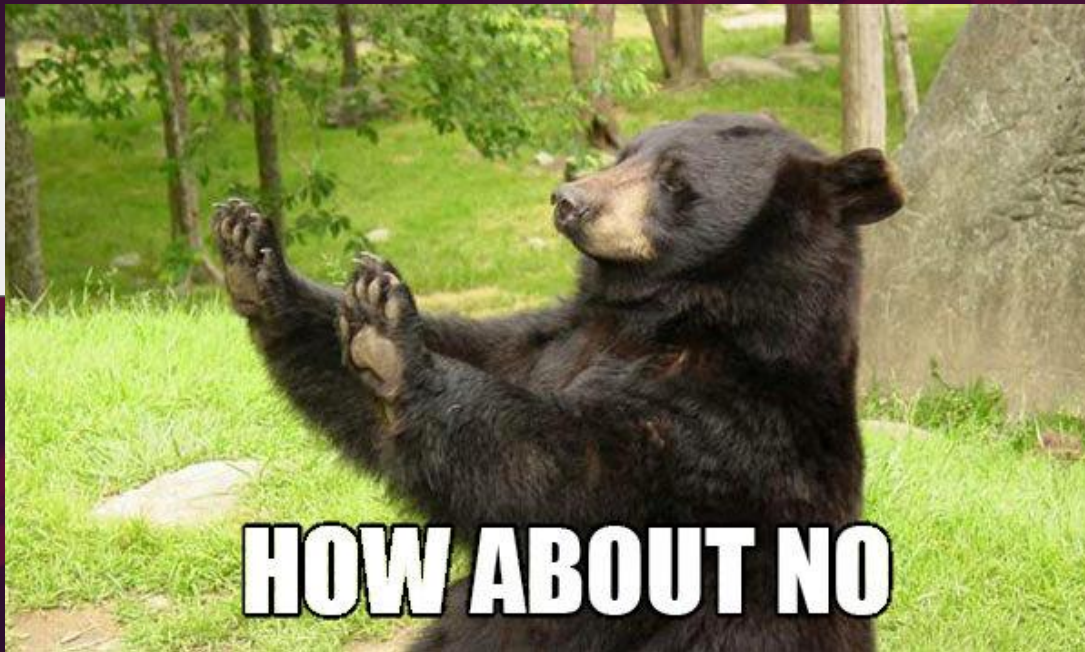
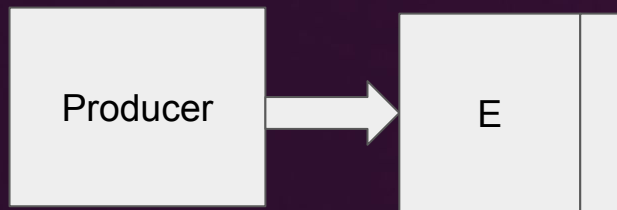
Схематично



Схематично



Схематично



Схематично



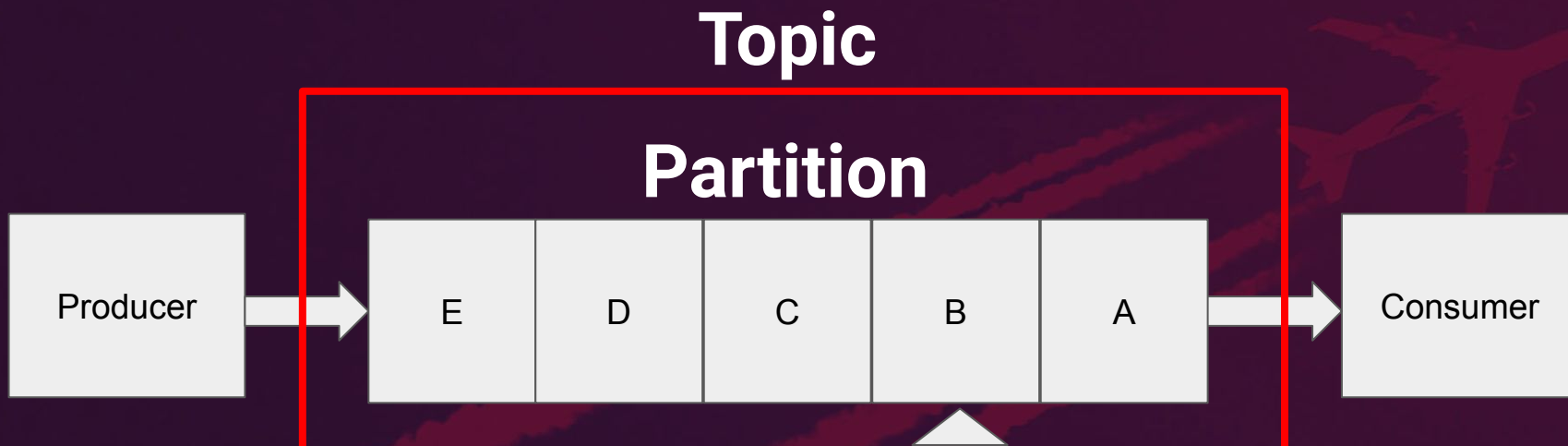
Схематично

Topic - тема, в которую пишут Producers

Partition - часть темы, которая может быть размещена на отдельном сервере. Минимальная распределяемая величина. По сути файл на диске.

Offset - указатель на последнее прочитанное сообщение

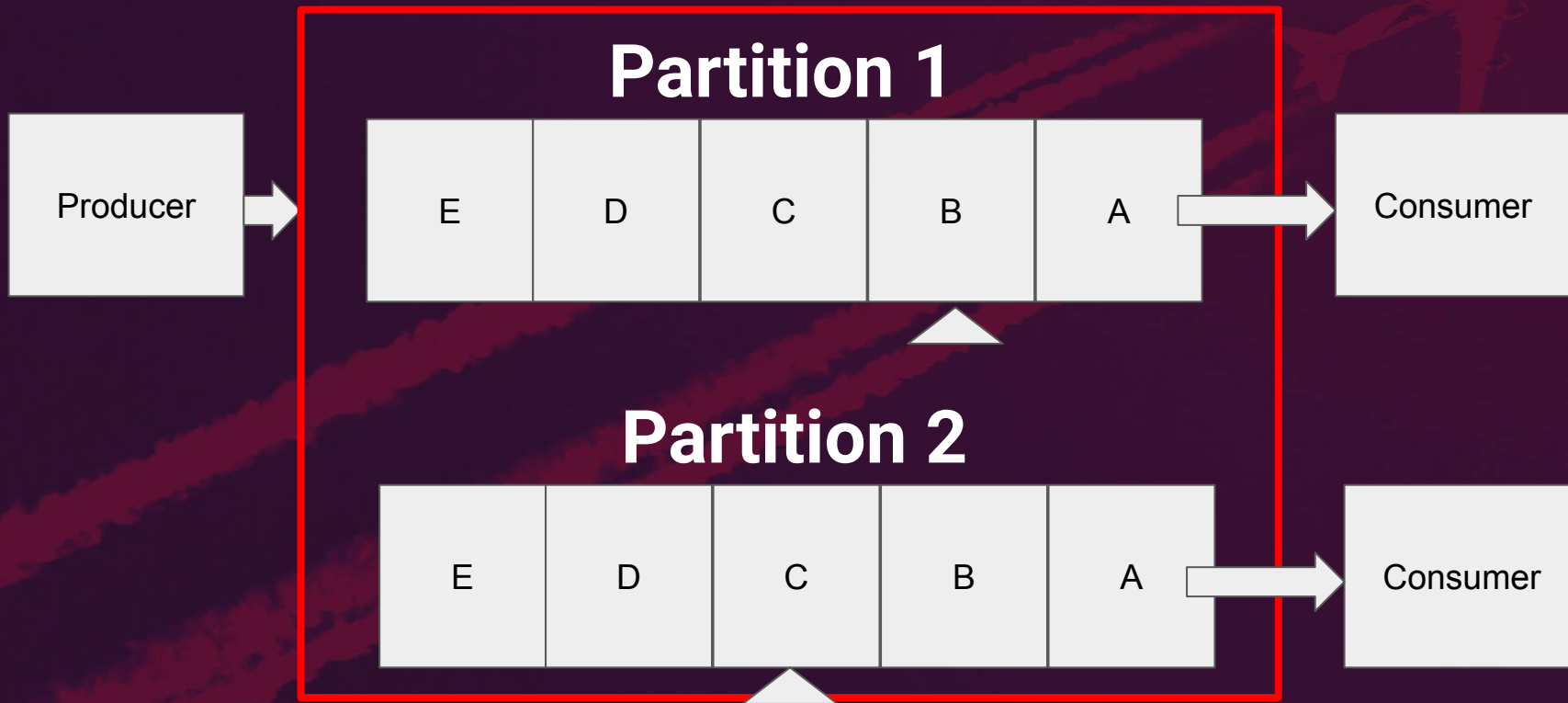
Схематично



1 Partition = 1 Consumer

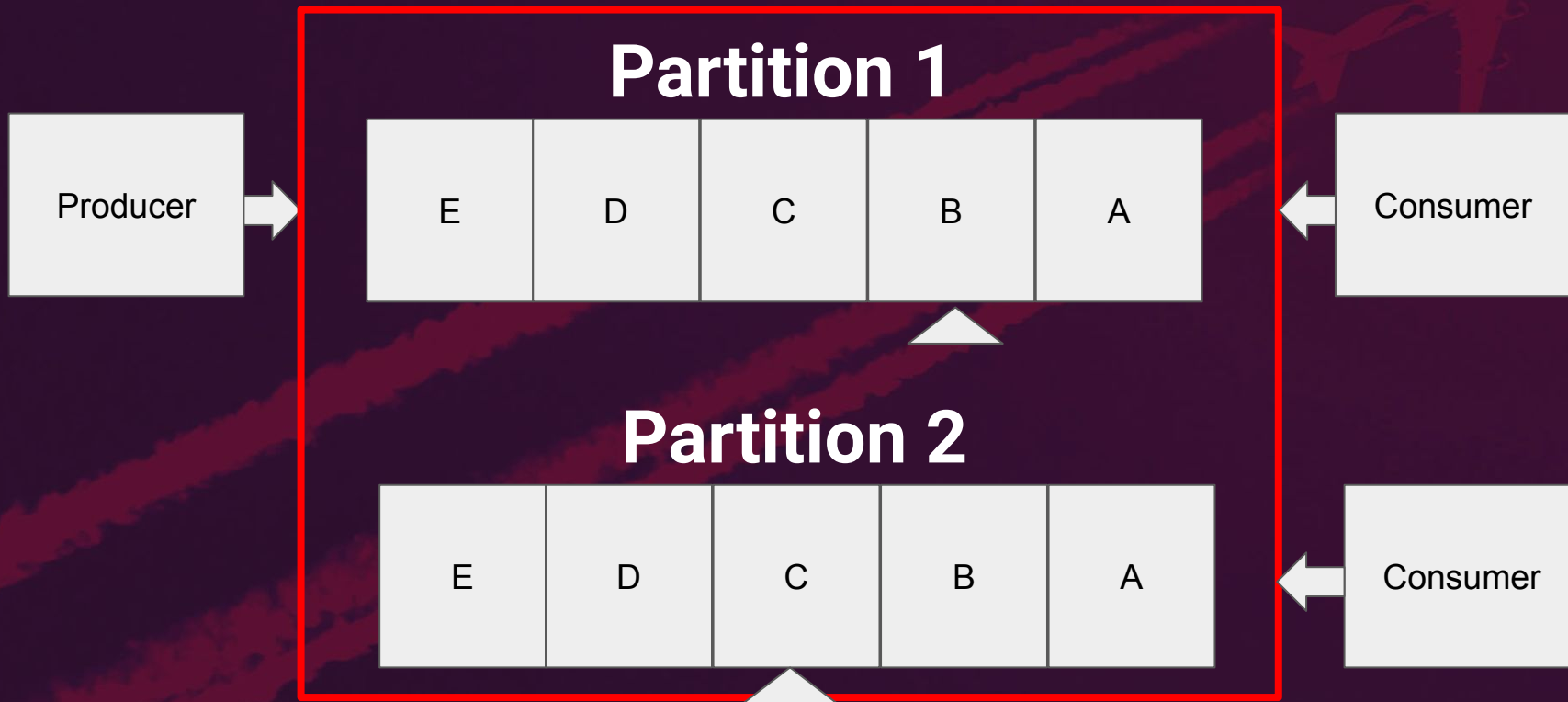
Схематично

Topic



Схематично

Topic



Фи́чи

- Kafka Streams / Kafka Tables
- Сдвиг offset и пере-проигрывание
- Сжатие лога
- ...



Redis



Что за зверь:

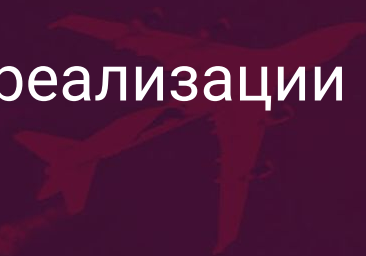
NoSQL Key-Value In-memory Database

Язык разработки: C

Основная задача

Используется как для баз данных, так и для реализации кэшей, брокеров сообщений

Хранит пары ключ-значение



Ключи

Описываются обычно паттерном:

х:у:z....

h?llo совпадает с **hello**, **hallo** и **hxllo**

h*llo совпадает с **hllo** и **heeeeello**

h[ae]llo совпадает с **hello** и **hallo**, но не **hillo**

h[^e]llo совпадает с **hallo**, и, ... но не **hello**

h[a-b]llo совпадает с **hallo** и **hblllo**



Типы значений

- Строки (могут использоваться так же как числа)
- Списки
- Множества
- Упорядоченные множества
- Хеши



Действия над строками

- Создать - SET
- Дописать в хвост - APPEND
- Посчитать длину строки - STRLEN
- Получить подстроку - GETRANGE
- Установить часть строки - SETRANGE

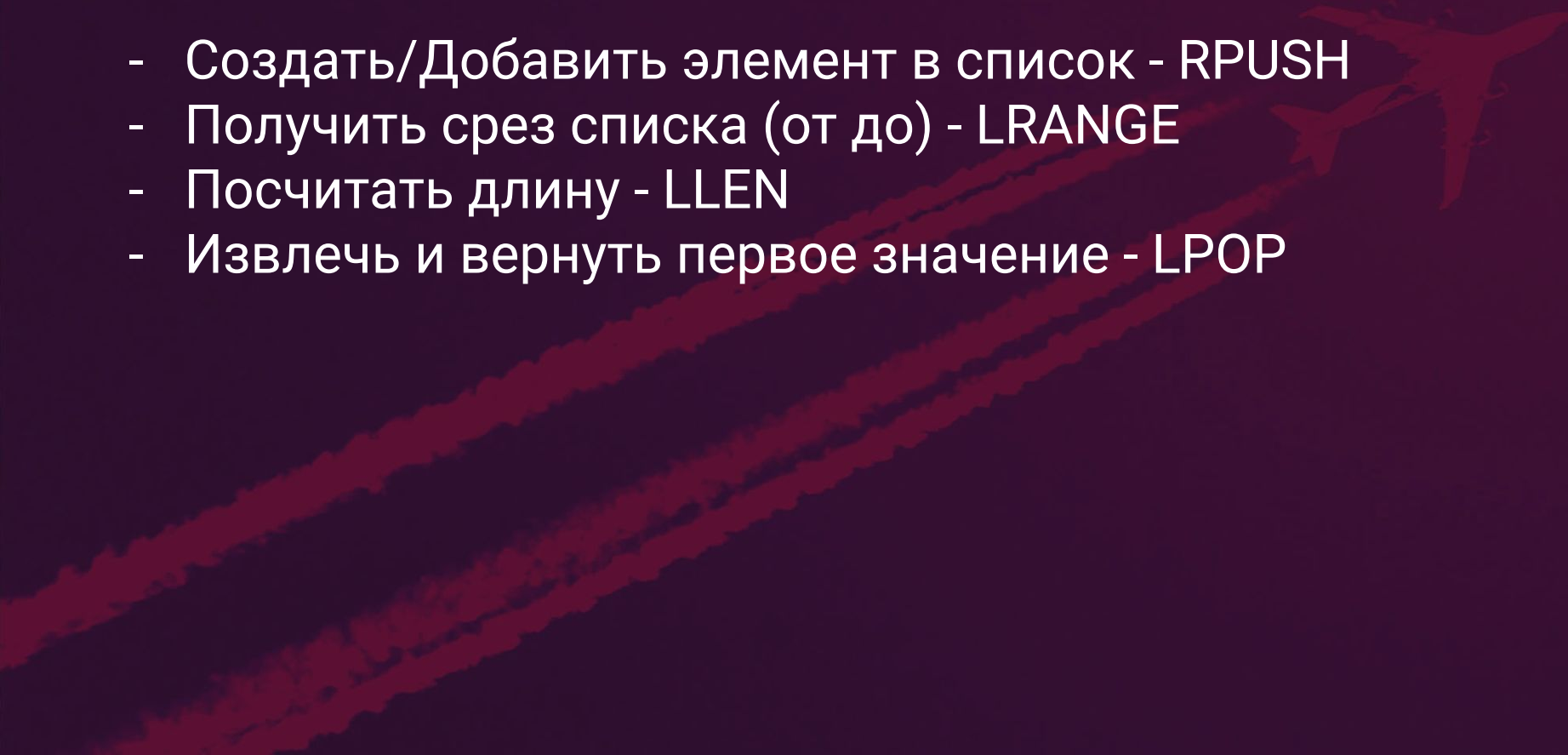


Действия над числами

- Создать - SET
- Увеличить на 1 - INCR
- Уменьшить на 1 - DECR
- Увеличить на N - INCRBY
- Уменьшить на N - DECRBY



Действия над списками

- Создать/Добавить элемент в список - RPUSH
 - Получить срез списка (от до) - LRANGE
 - Посчитать длину - LLEN
 - Извлечь и вернуть первое значение - LPOP
- 

Действия над множествами, хешами...

- Добавления
- Удаления
- Сравнения, разница и так далее

Смотрим мануал :)



Фи́чи

- Транзакции
- Expire
- Publish / Subscribe
- ...



Memcached



Что за зверь:

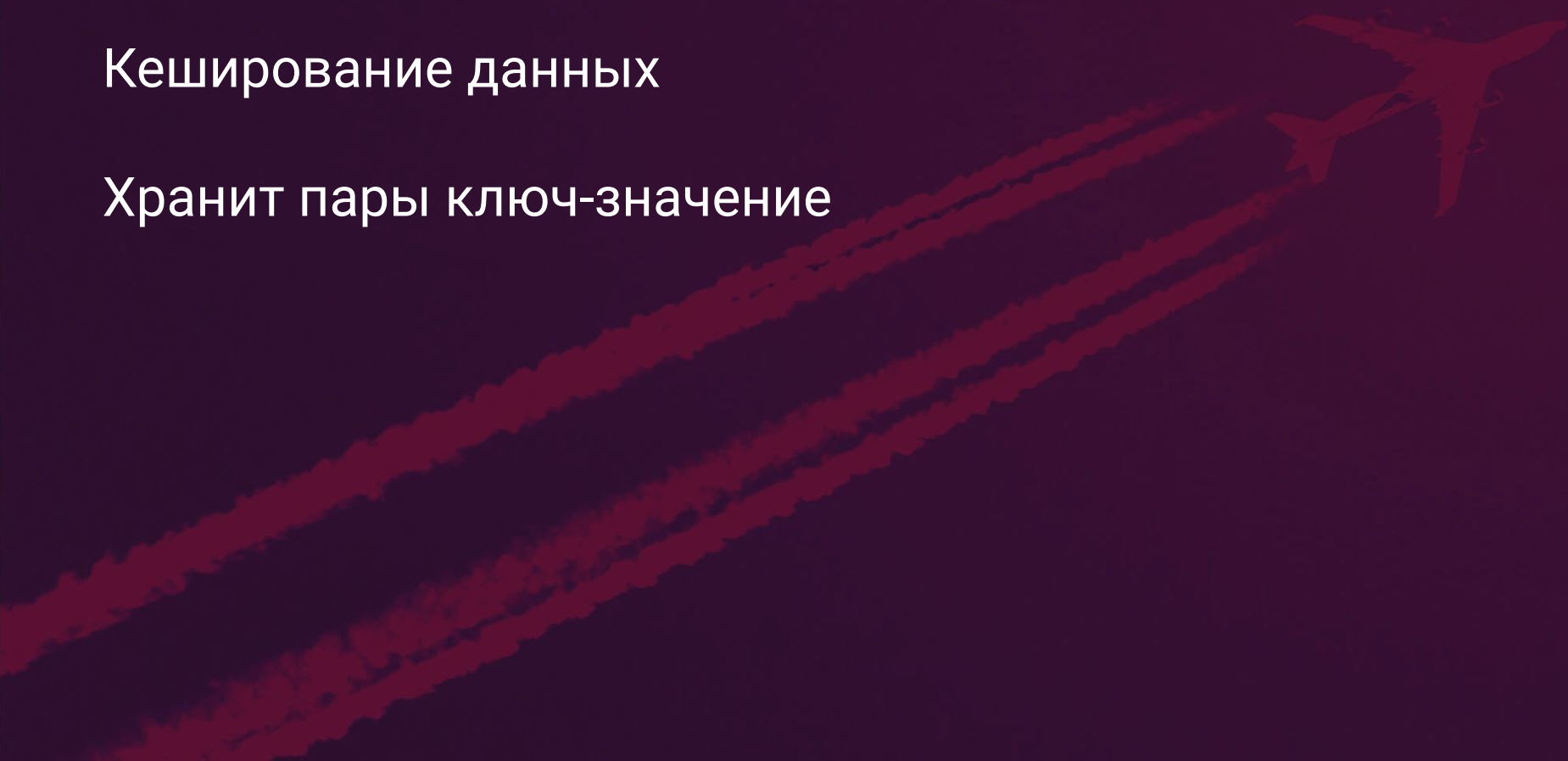
сервис кэширования данных в оперативной памяти на основе хеш-таблицы (ключ-значение)

Язык разработки: C

Основная задача

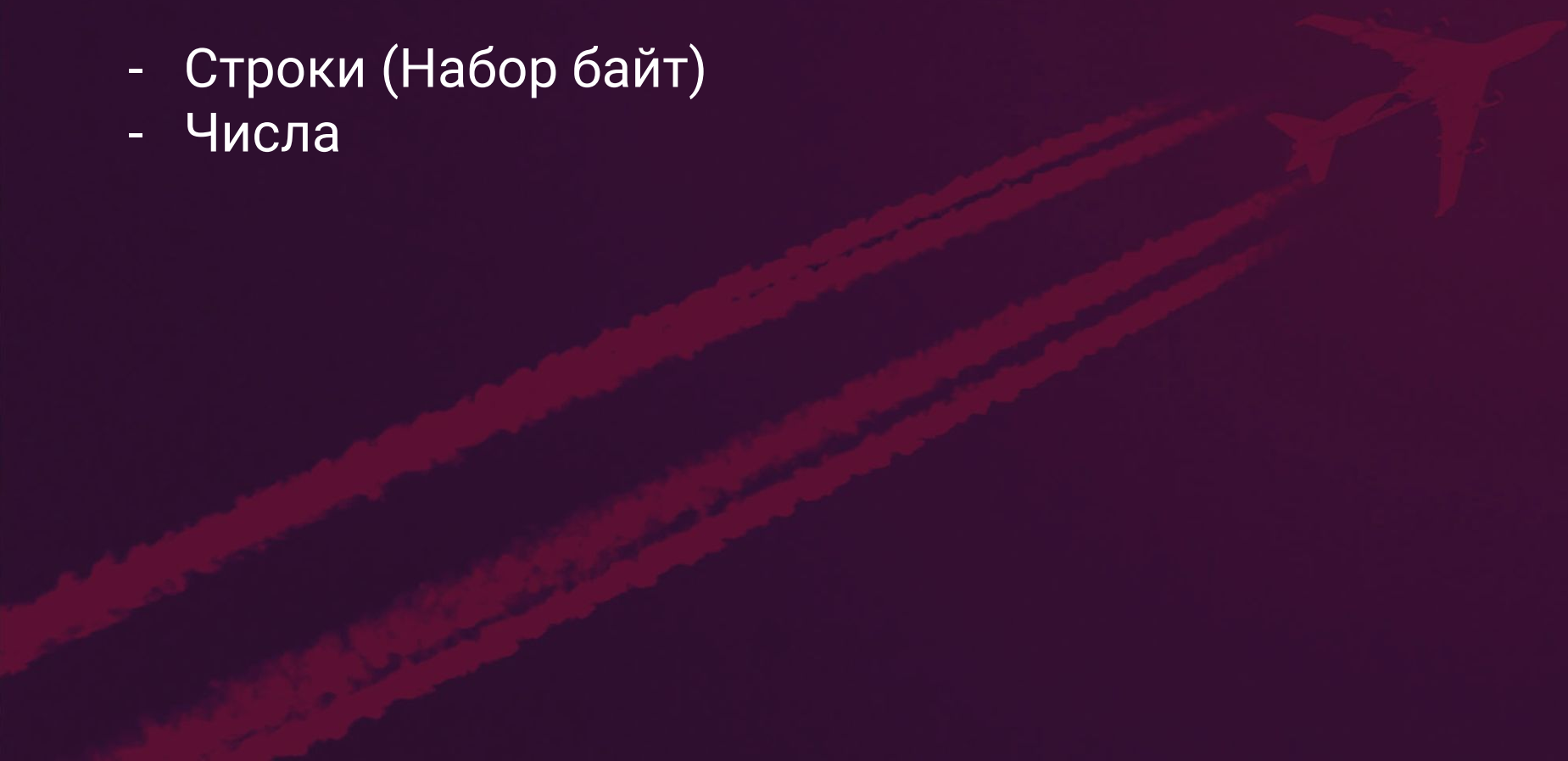
Кеширование данных

Хранит пары ключ-значение

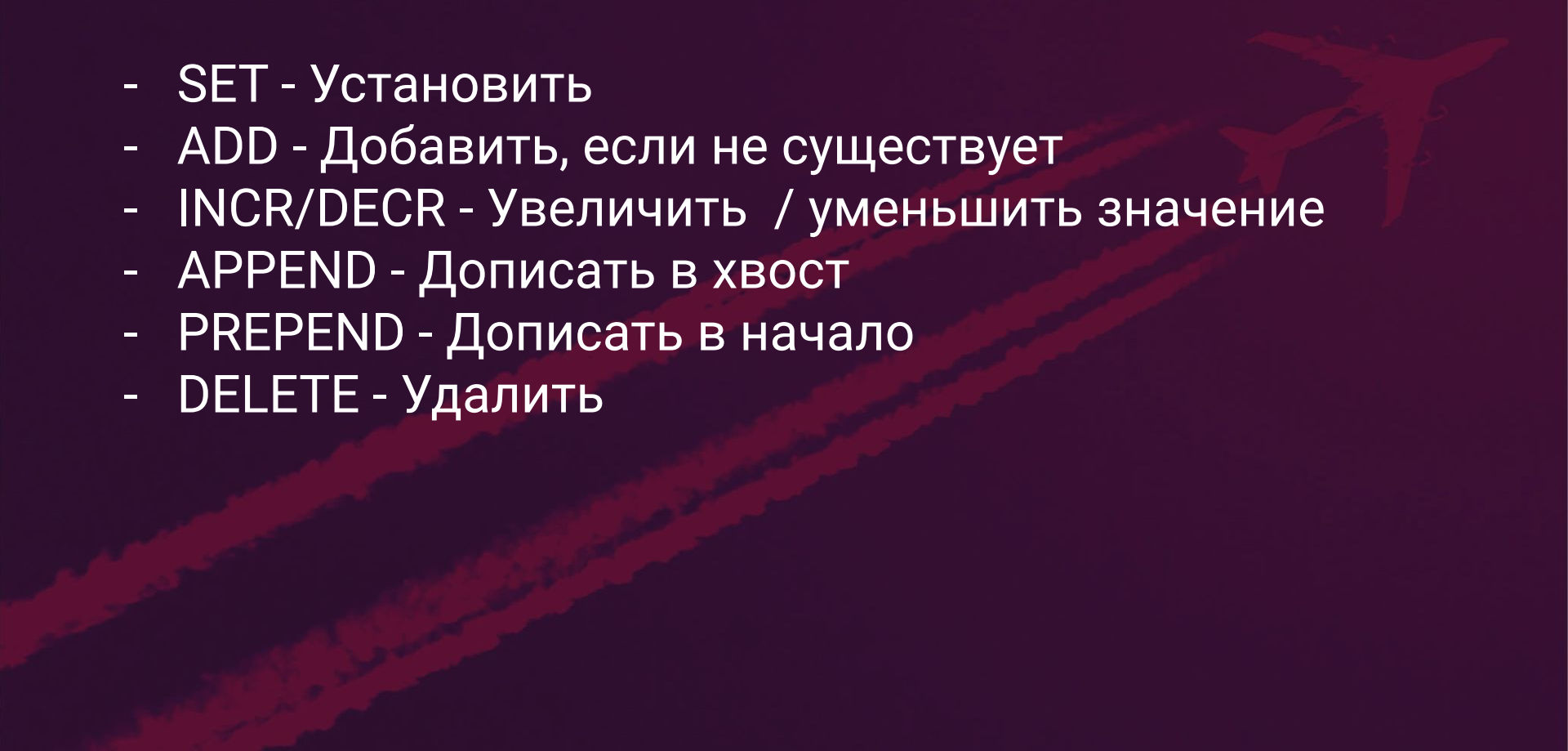


Типы значений

- Строки (Набор байт)
- Числа



Действия

- SET - Установить
 - ADD - Добавить, если не существует
 - INCR/DECR - Увеличить / уменьшить значение
 - APPEND - Дописать в хвост
 - PREPEND - Дописать в начало
 - DELETE - Удалить
- 

Особенности

- Всё всегда в RAM
- Удаление данных
 - Истекло время жизни
 - Не хватило памяти и обращений по данному ключу не так много
 - Сервер упал
- Архитектура рассчитана на максимальную производительность



NuxtJS

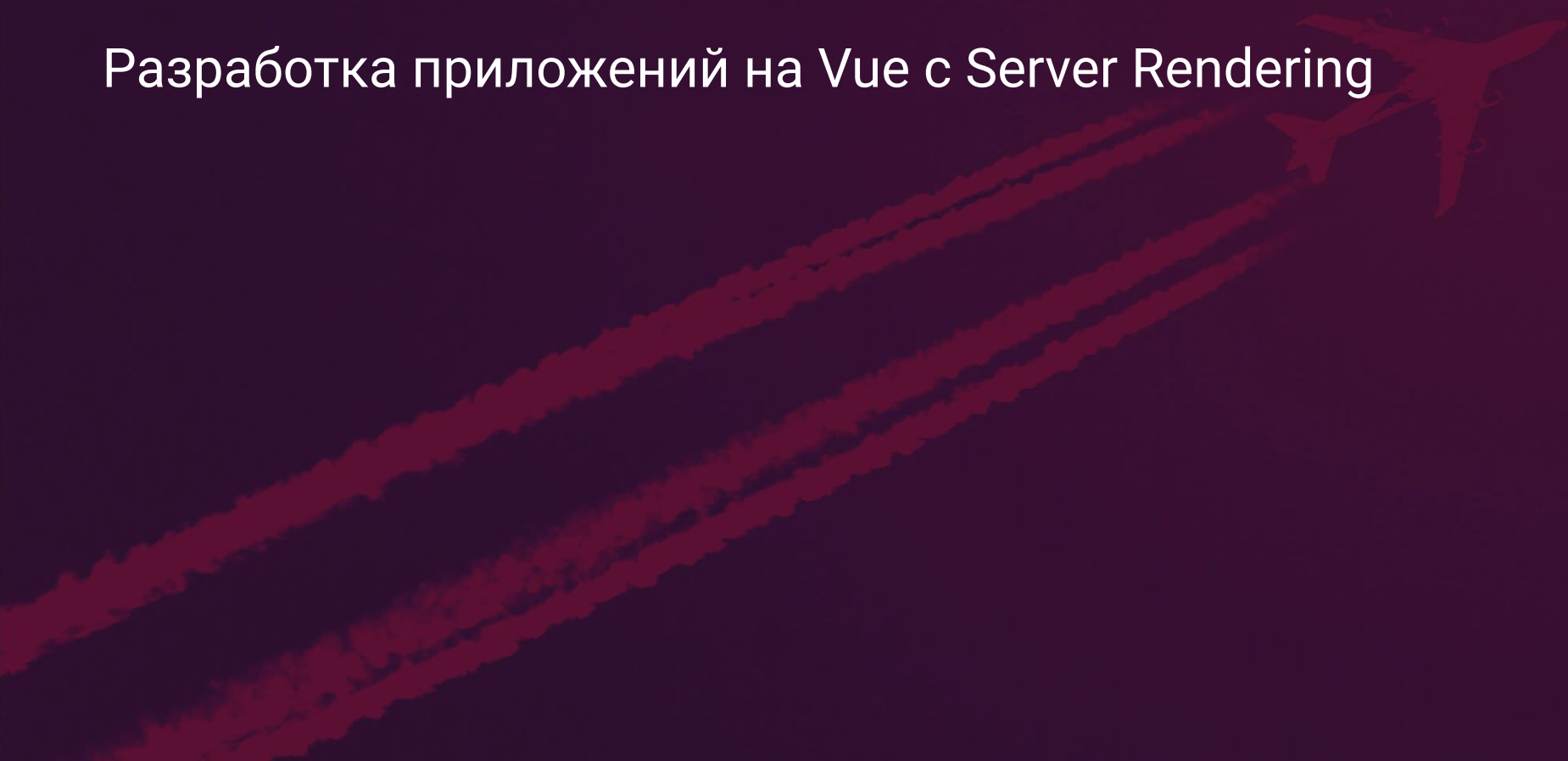


Что за зверь:
фреймворк для создания приложений на Vue.js

Язык разработки: JS

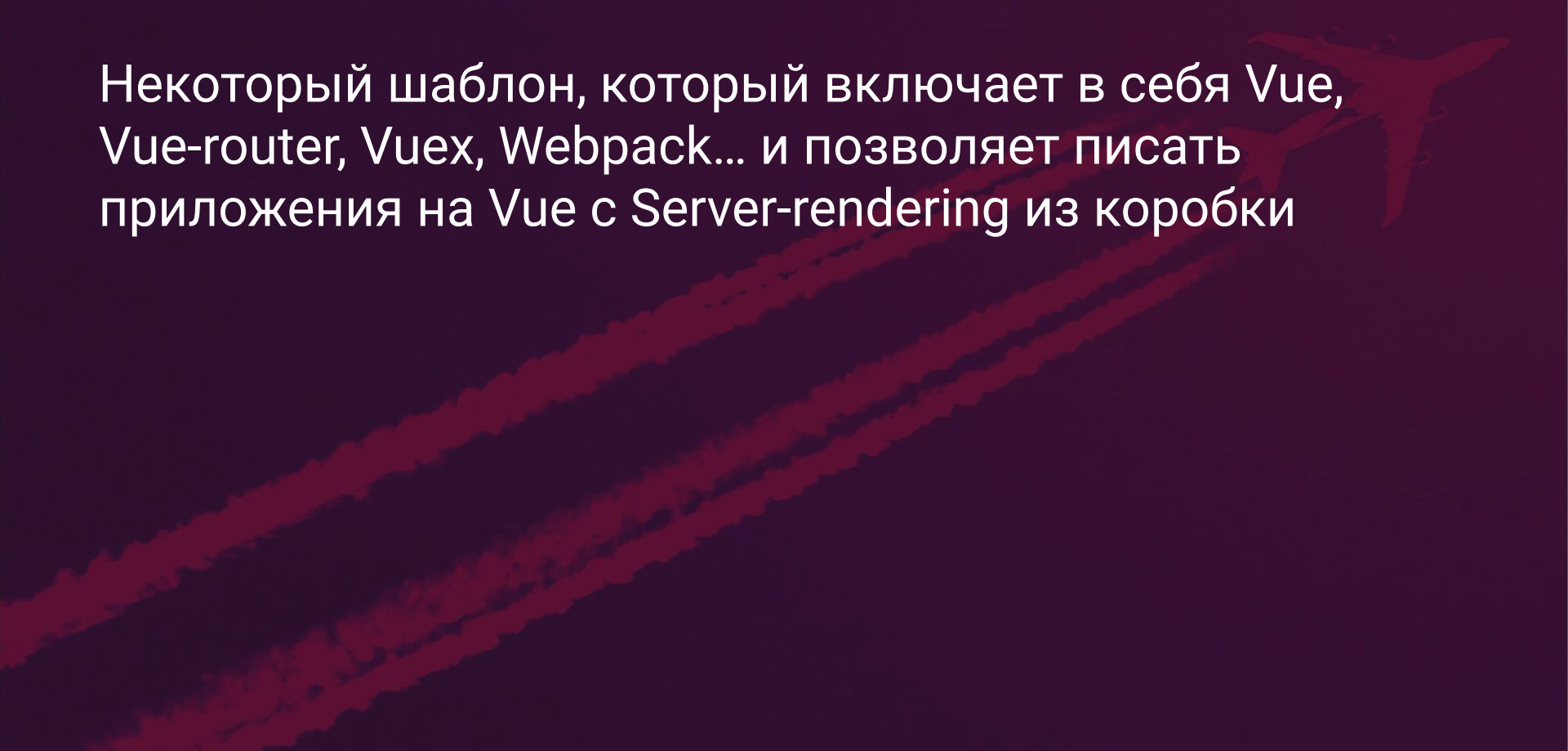
Основная задача

Разработка приложений на Vue с Server Rendering

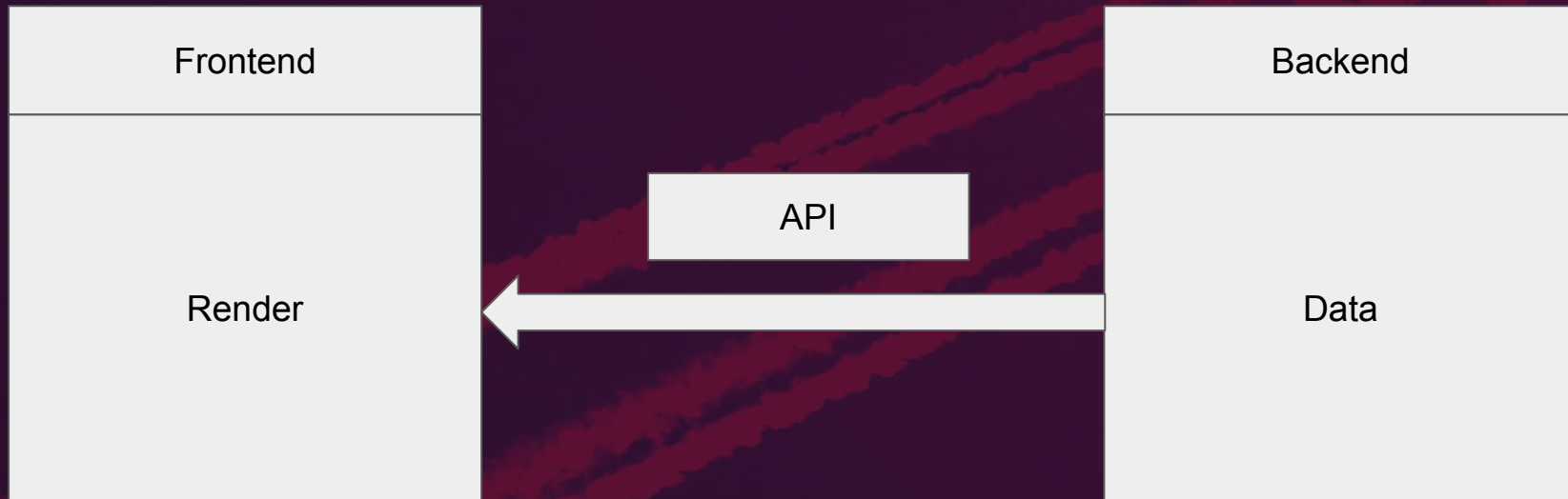


По сути своей

Некоторый шаблон, который включает в себя Vue, Vue-router, Vuex, Webpack... и позволяет писать приложения на Vue с Server-rendering из коробки

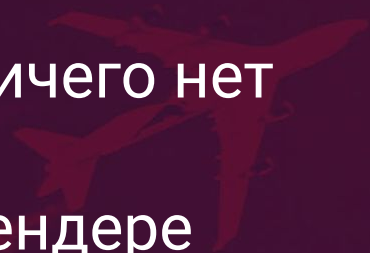


Client Rendering: cxema



Client Rendering: минусы

- Если сдох/отключен/не догрузился JS - ничего нет
- Время до первой отрисовки
- Выше нагрузка на клиент в при первом рендере

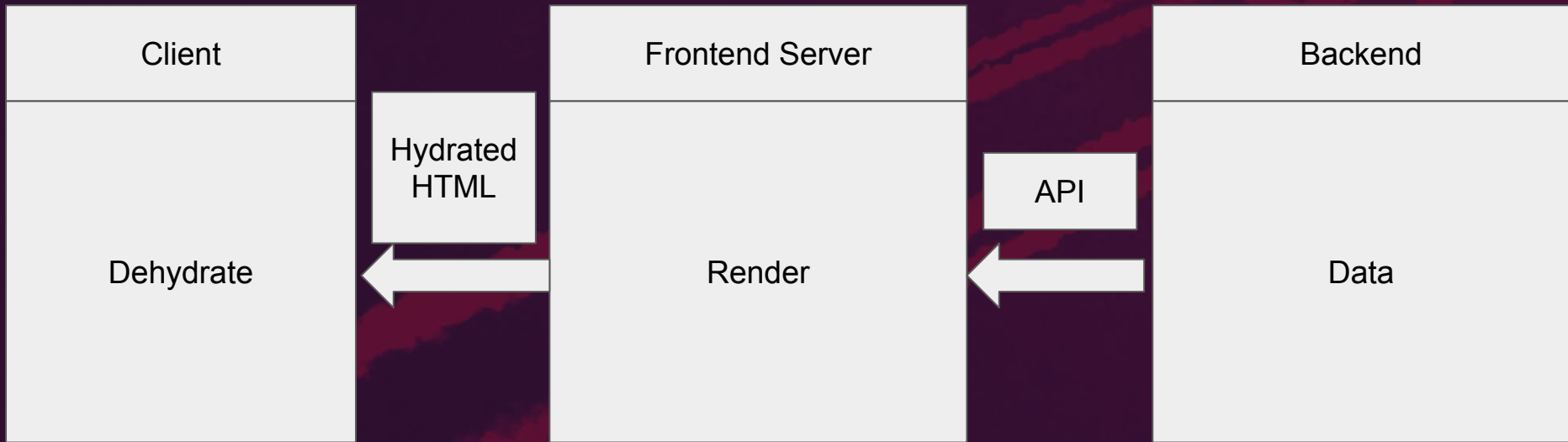


Client Rendering: минусы

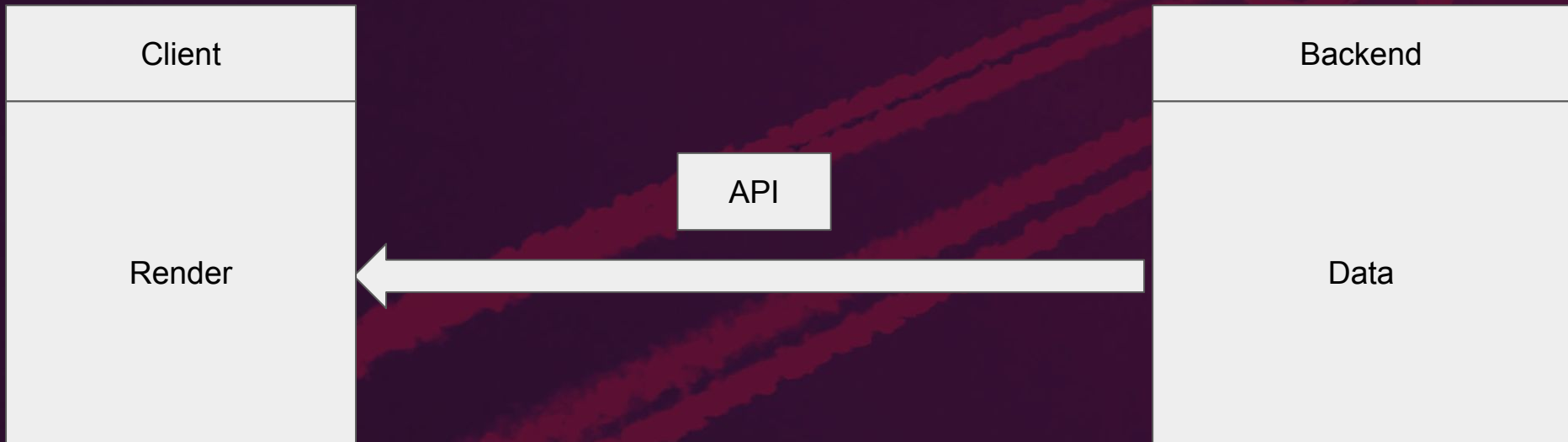
- Если сдох/отключен/не догрузился JS - ничего нет
- Время до первой отрисовки
- Выше нагрузка на клиент в при первом рендере

- SEO

Server Rendering: схема первого рендера



Server Rendering: схема взаимодействия



Server Rendering: минусы

- Нагрузка на сервер
- Новая пачка технологий
- Некоторые ограничения при разработке Vue-приложения



MongoDB



Что за зверь:

документоориентированная система управления базами данных

Язык разработки: C, C++, JS

Основная задача

Хранение данных.

Отсутствие схемы данных.

No SQL.

JSON-подобные документы.



Вставка данных

```
db.band.insert({  
  name: 'Queen',  
  bid: '3'  
});
```

Обновление данных

```
db.band.update({
  bid: '1'
}, {
  $set: { // Тут могут быть: $set, $unset, $inc, $rename
    members: [
      {name: "Jimmy Page", id: "1"},
      {name: "Robert Plant", id: "2"},
      {name: "John Bonham", id: "3"},
      {name: "John Paul Jones", id: "4"}
    ]
  }
})
```

Поиск данных

```
db.band.find({  
  name: 'Queen'  
});
```

```
db.band.find({  
  id: {$gt: 10} // $lt, $lte, $gt, $gte, $ne  
}, {  
  name: 1 // Выбрать только имена  
});
```

```
db.band.find({  
  $where: function() {  
    return (hex_md5(this.name) == "9b53e667f30cd329dca1ec9e6a83e994")  
  } // Выполнить JS на сервере  
});
```

Chain methods

```
db.unicorns.find({}, {name: true}).sort({name: -1}).limit(3).skip(1)
```

```
db.band.count({age: {$gt: 50}})
```

Фичи

- Простое масштабирование
- MapReduce
- Отсутствие классических транзакций
- Вложенные документы



PostgreSQL



Что за зверь:

свободная объектно-реляционная система
управления базами данных

Язык разработки: C

Поля-массивы

```
CREATE TABLE holiday_picnic (  
    holiday varchar(50) -- строковое значение  
    sandwich text[], -- массив  
    side text[] [], -- многомерный массив  
    dessert text ARRAY, -- массив  
    beverage text ARRAY[4] -- массив из 4-х элементов  
);
```


JSONB

```
CREATE TABLE json_test (  
  id serial primary key,  
  data jsonb  
);
```

```
INSERT INTO json_test (data) VALUES  
( '{}'),  
( '{"a": 1}'),  
( '{"a": 2, "b": ["c", "d"]}'),  
( '{"a": 1, "b": {"c": "d", "e": true}}'),  
( '{"b": 2}');
```

```
SELECT * FROM json_test WHERE data ->> 'a' > '1';  
-- Отдаст: ( '{"a": 2, "b": ["c", "d"]}')
```

Собственные типы

```
-- создаем новый составной тип "wine"
CREATE TYPE wine AS (
    wine_vineyard varchar(50),
    wine_type varchar(50),
    wine_year int
);

-- создаем таблицу, которая использует составной тип "wine"
CREATE TABLE pairings (
    menu_entree varchar(50),
    wine_pairing wine
);

-- вставляем данные в таблицу при помощи выражения ROW
INSERT INTO pairings VALUES
    ('Lobster Tail', ROW('Stag's Leap', 'Chardonnay', 2012)),
    ('Elk Medallions', ROW('Rombauer', 'Cabernet Sauvignon', 2012));

SELECT (wine_pairing).wine_vineyard, (wine_pairing).wine_type
FROM pairings
WHERE menu_entree = 'Elk Medallions';
```

Фи́чи

- Меньше ограничений по размеру
- Более гибкое масштабирование по сравнению с MySQL
- Больше встроенных типов данных
- Удобная работа с геометрическими данными
- Огромные возможности для индексации
- Огромные возможности для расширения
- Рекурсивные запросы
- Материализованные представления
- ...



Bcë :)

