QIWI

6 причин подумать, прежде чем использовать WebSocket

Ноябрь 2023

Михаил Житков

Работаю в QIWI, команда Qchat

Почему вообще Websocket

```python
from fastapi import FastAPI, WebSocket
from fastapi.responses import PlainTextResponse

app = FastAPI()

@app.get("/")
async def root():
    return PlainTextResponse("Hello world")

@app.websocket("/ws")
async def websocket_route(websocket: WebSocket):
    await websocket.accept()
    while True:
        data = await websocket.receive_text()
        await websocket.send_text(f"Message text was: {data}")
```

https://www.pubnub.com/learn/glossary/what-are-server-sent-events

Для кого этот доклад

Из чего состоит доклад

1. **Вебсокеты в теории**
2. **Вебсокеты в наших реалиях**
3. **Шишки (те самые причины подумать)**
    a. Другая парадигма веб-приложения
    b. Надо по-другому разрабатывать
    c. Какие проблемы вызывает масштабирование
    d. Аутентификация
    e. О чем не думают бэкендеры
    f. Как жить без OpenAPI

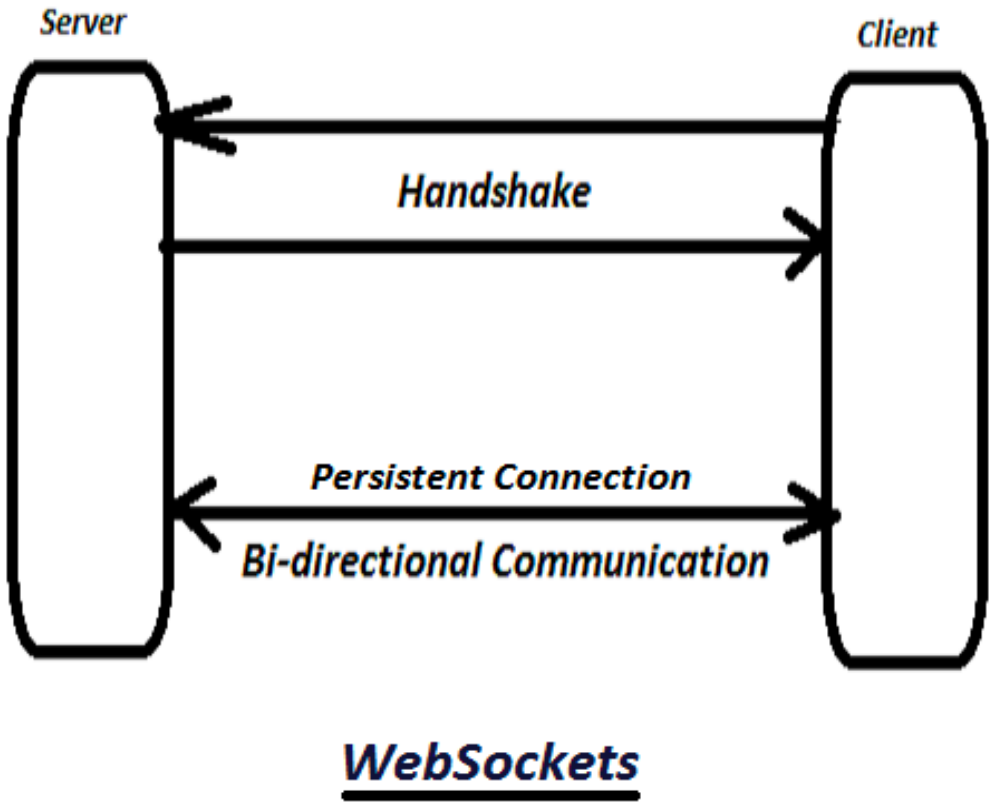
01

Вебсокеты в теории

Вебсокеты

Протокол WebSocket был стандартизирован IETF в RFC 6455 в 2011 году

Поддерживается обмен текстовыми или бинарными данными



Сам RFC:

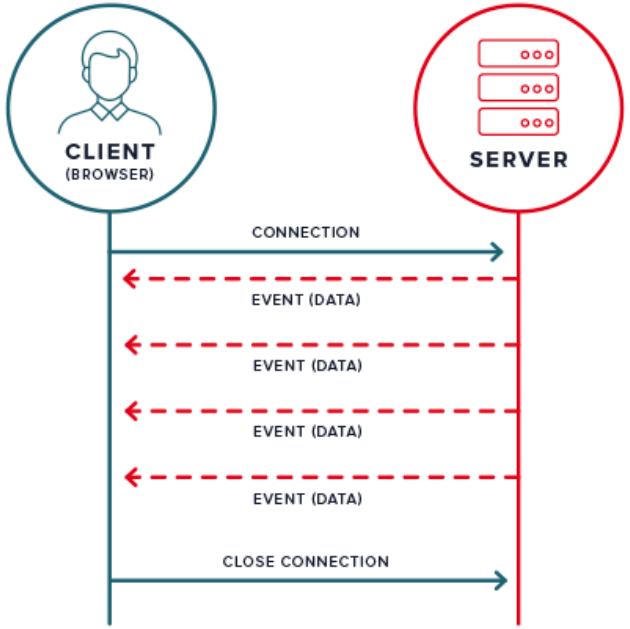
https://datatracker.ietf.org/doc/html/rfc6455

https://images.techhive.com/images/article/2016/06/websockets-100668229-primary.idge.jpg

Похожие технологии

Server Side Events

Концептуальная схема



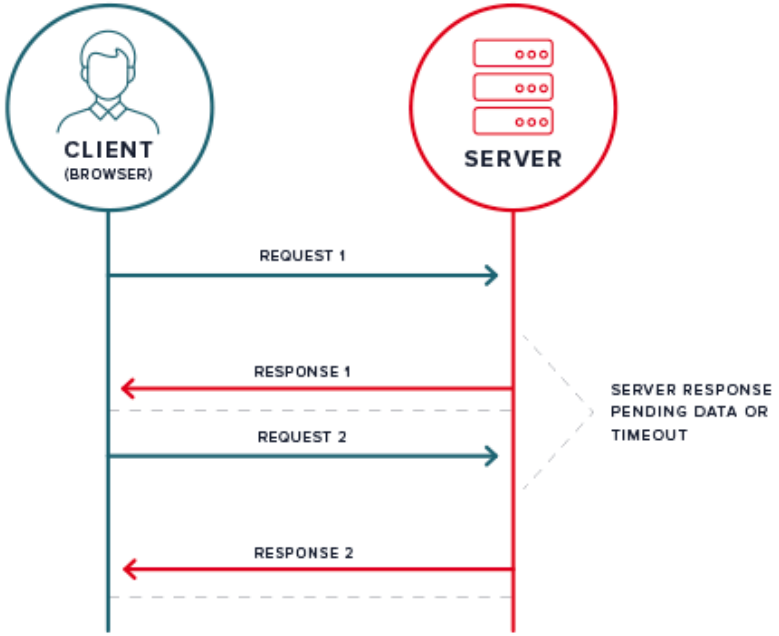
Отличие от WebSockets

- События только с сервера

https://www.pubnub.com/learn/glossary/what-are-server-sent-events

Short/Long Polling

Концептуальная схема



Отличие от WebSockets

- Более “рваная” коммуникация
- HTTP
- Частые запросы

https://www.pubnub.com/blog/http-long-polling/

02

Вебсокеты в наших реалиях

Требования к “серьёзной” разработке

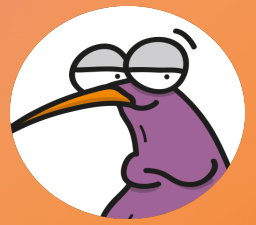
1. Отказоустойчивость
2. Масштабируемость
3. Безопасность
4. Иные, но в отдельном докладе

03

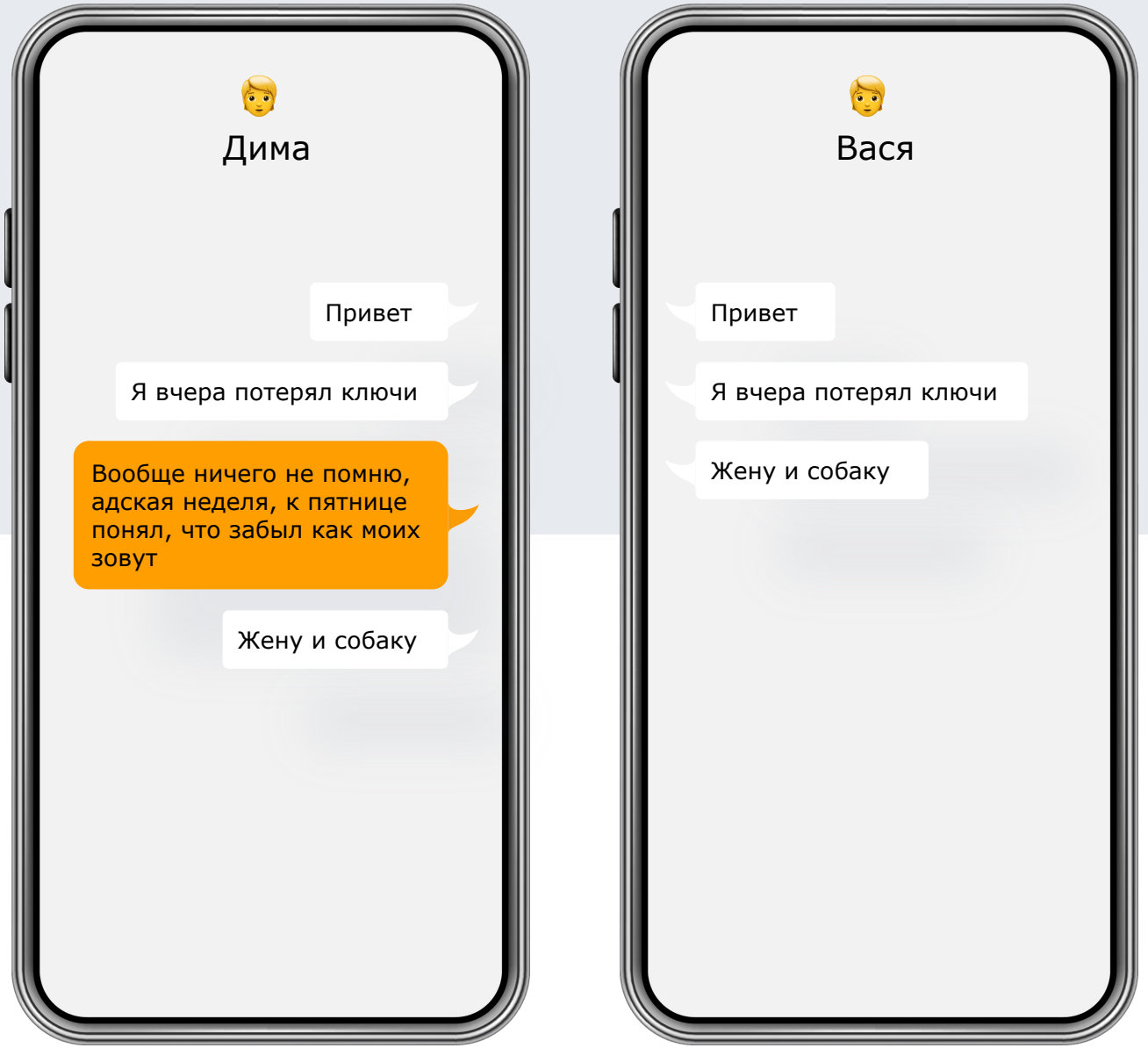
Шишки

Другая парадигма веб-сервиса

Клиентам важно иметь согласованное состояние



Почему клиентское состояние важно



Правило № 1

Следите за согласованностью
клиентского состояния

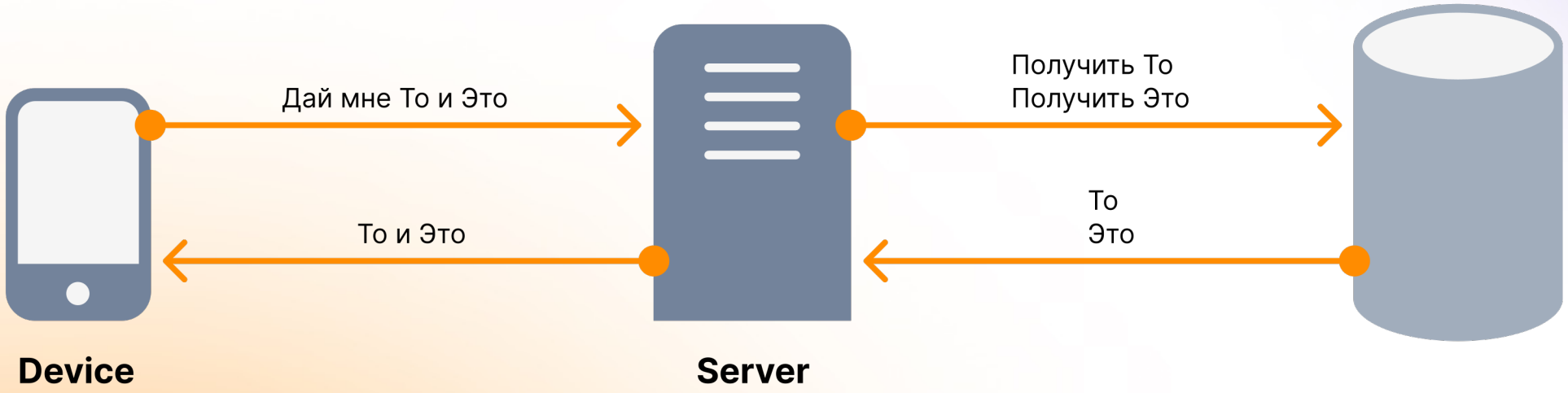
Состояние приложения

Stateful	Stateless
Часть состояния клиента лежит на сервере	Состояние клиента лежит на клиенте, включая способы актуализации состояния
Один сервер небыстро заменить другим	Сервера взаимозаменяемы в любой момент
Вывод сервера из строя вынуждает клиент заново создавать сессию	Система более масштабируема и устойчива к отказам

Как восстанавливать состояние

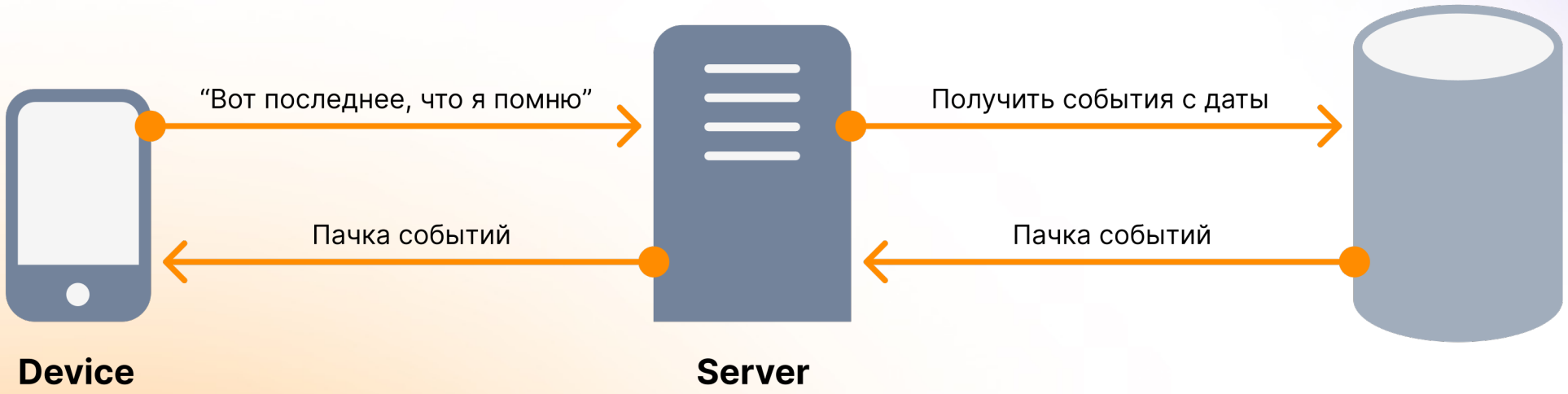
Pull-модель

Клиент знает, как получать нужные данные

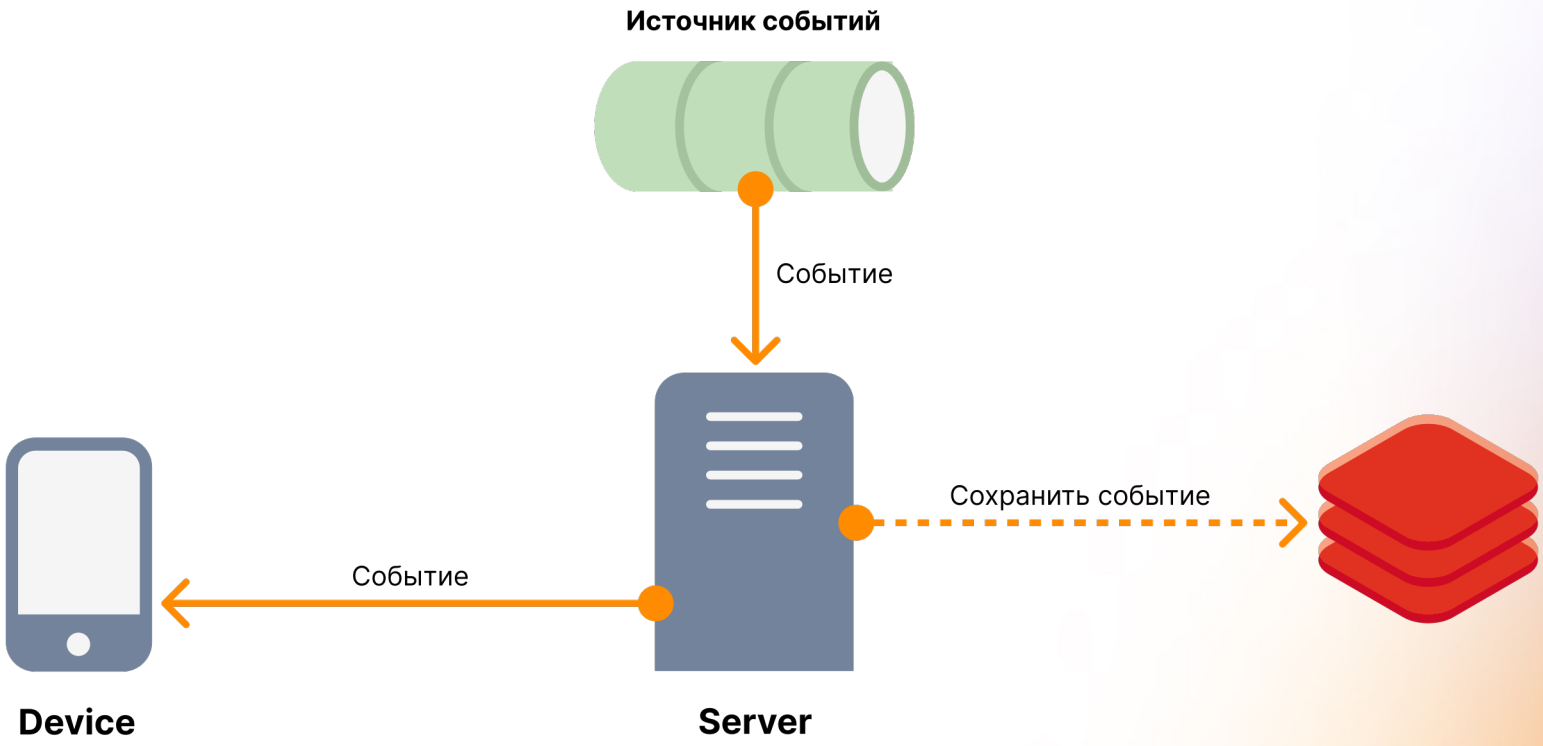


Push-модель

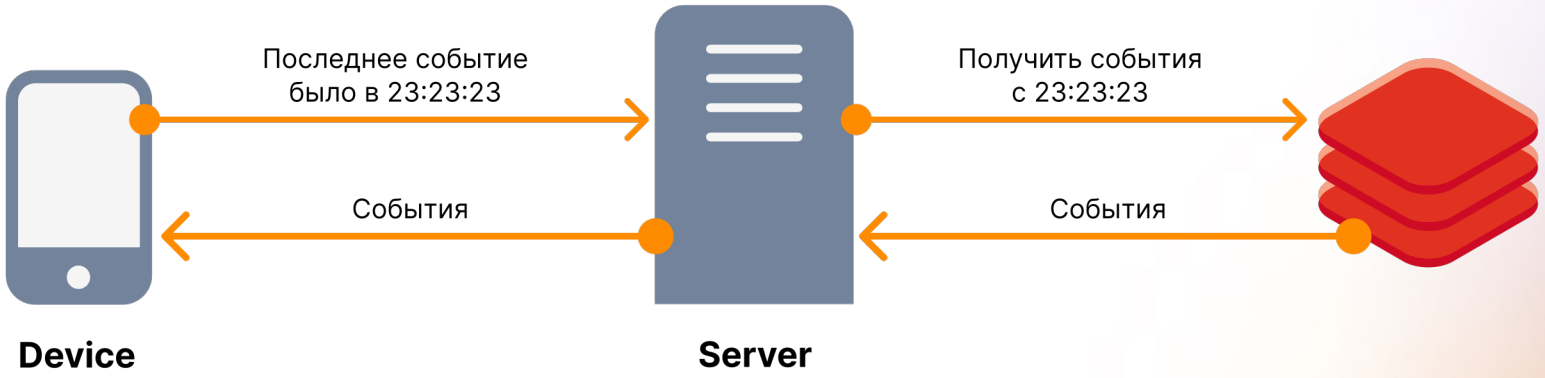
Push-модель позволяет сделать клиент
с меньшим количеством логики



Push-модель: как делать



Push-модель: как делать



Чего это будет стоить

N * S * Q(t)

где:

N - среднее число соединений

S - средний размер сообщения

Q - среднее количество сообщения в буфере
в настроенный промежуток времени

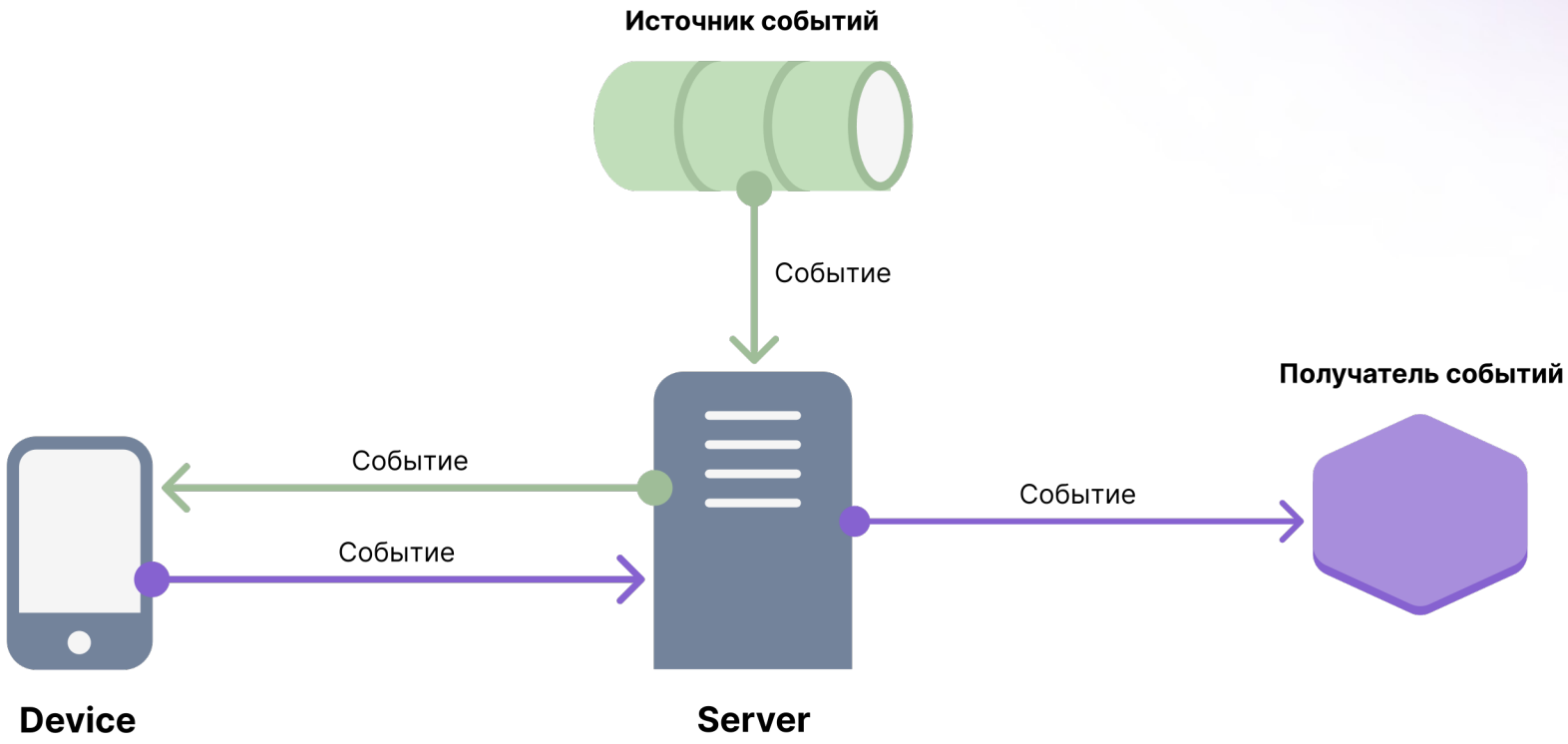
t - настроенный промежуток времени для буферизации

Надо по-другому разрабатывать

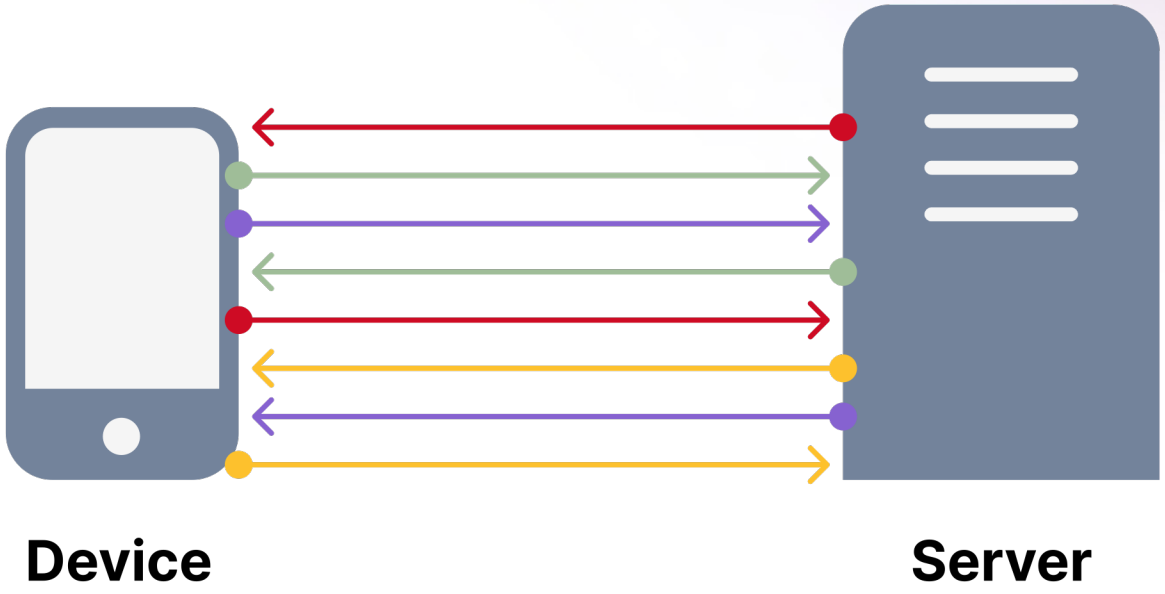
Упрощенная схема взаимодействия

Источник и получатель событий – логические.

Это могут быть как другие сервисы и компоненты, так и модуль на том же сервере



Взаимодействие внутри вебсокета неупорядоченное



Правило № 2

Не используйте Websocket для обмена событиями, которым требуется подтверждение обработки

Проблемы масштабирования

Как оно обычно работает

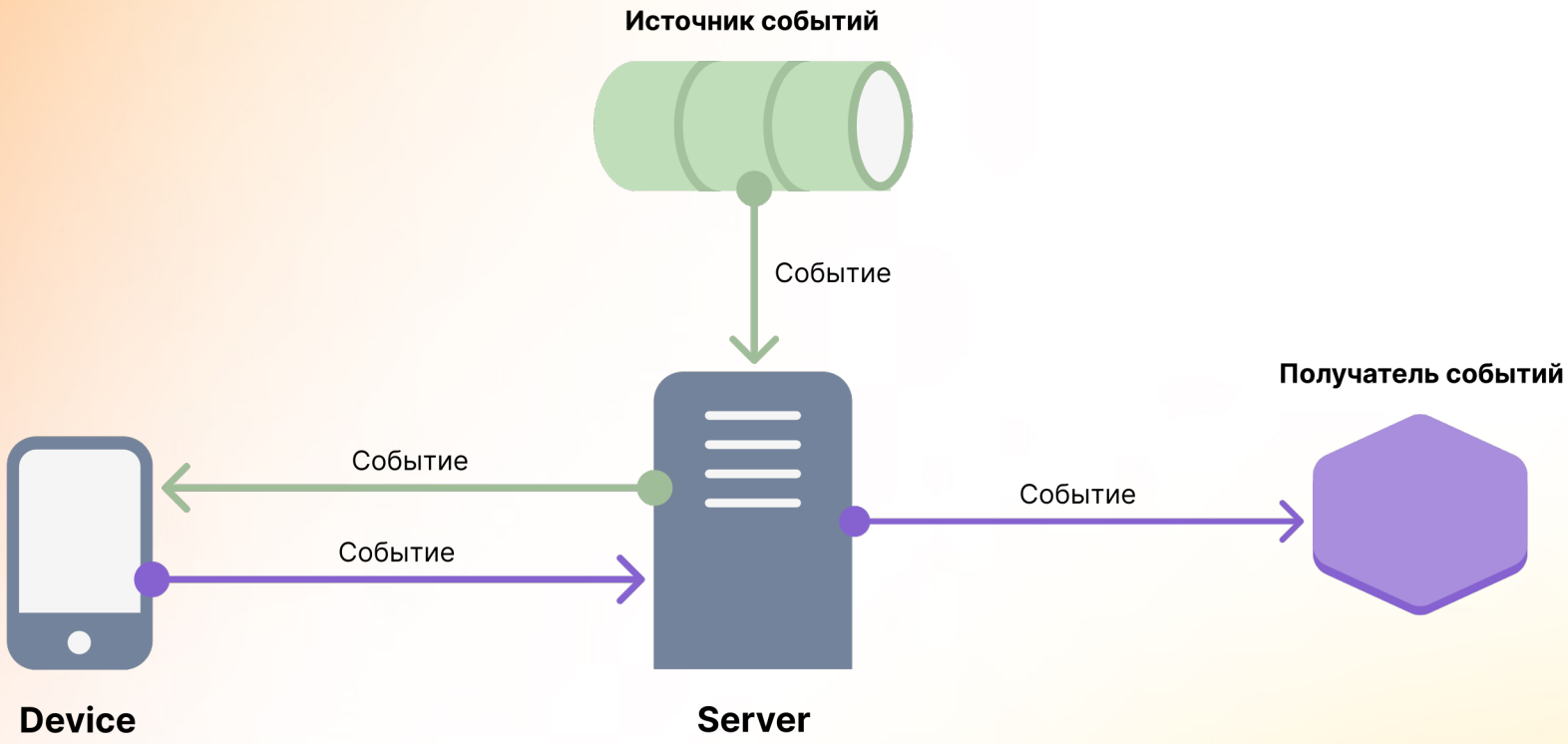
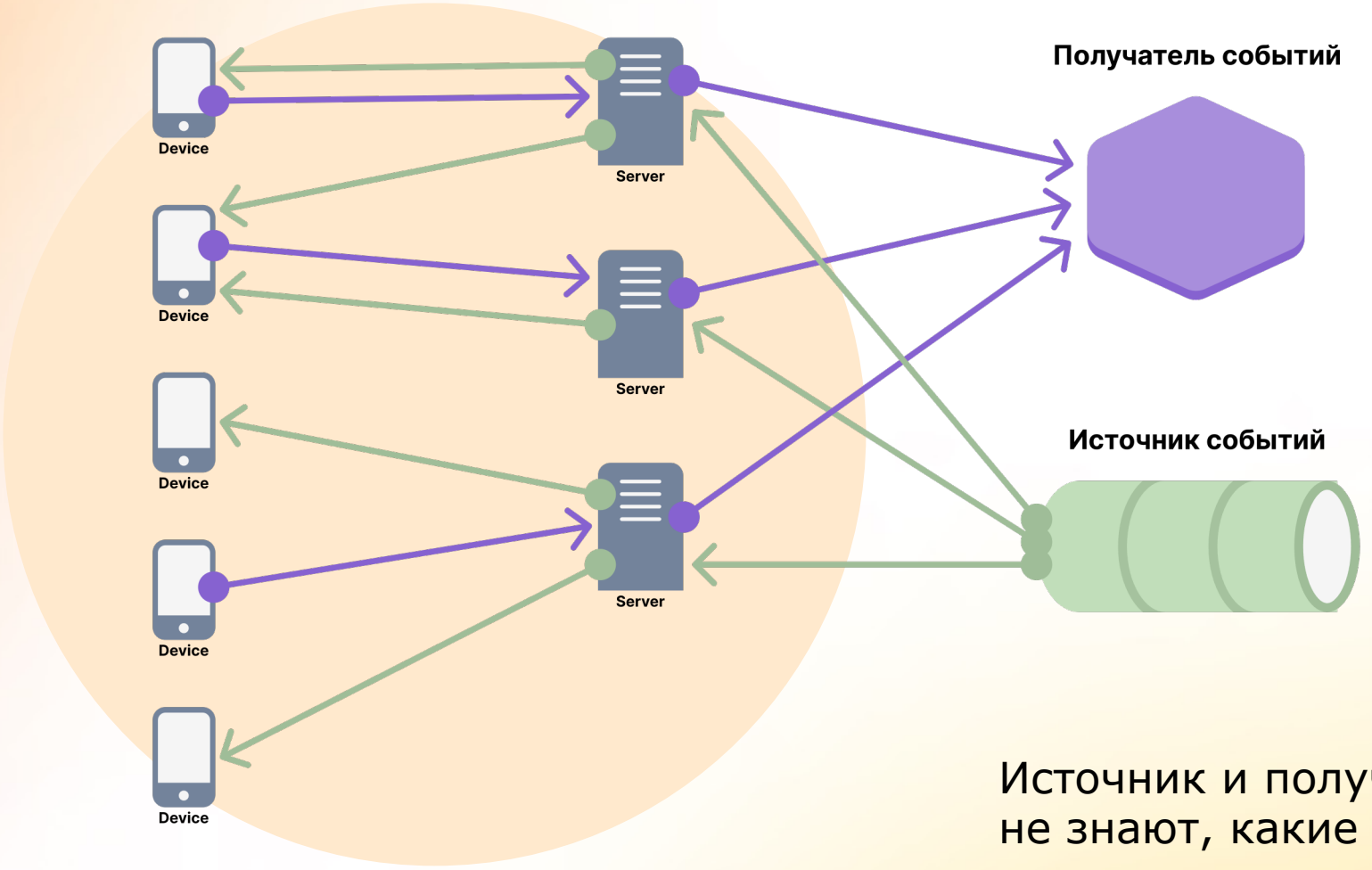


Схема в реальности

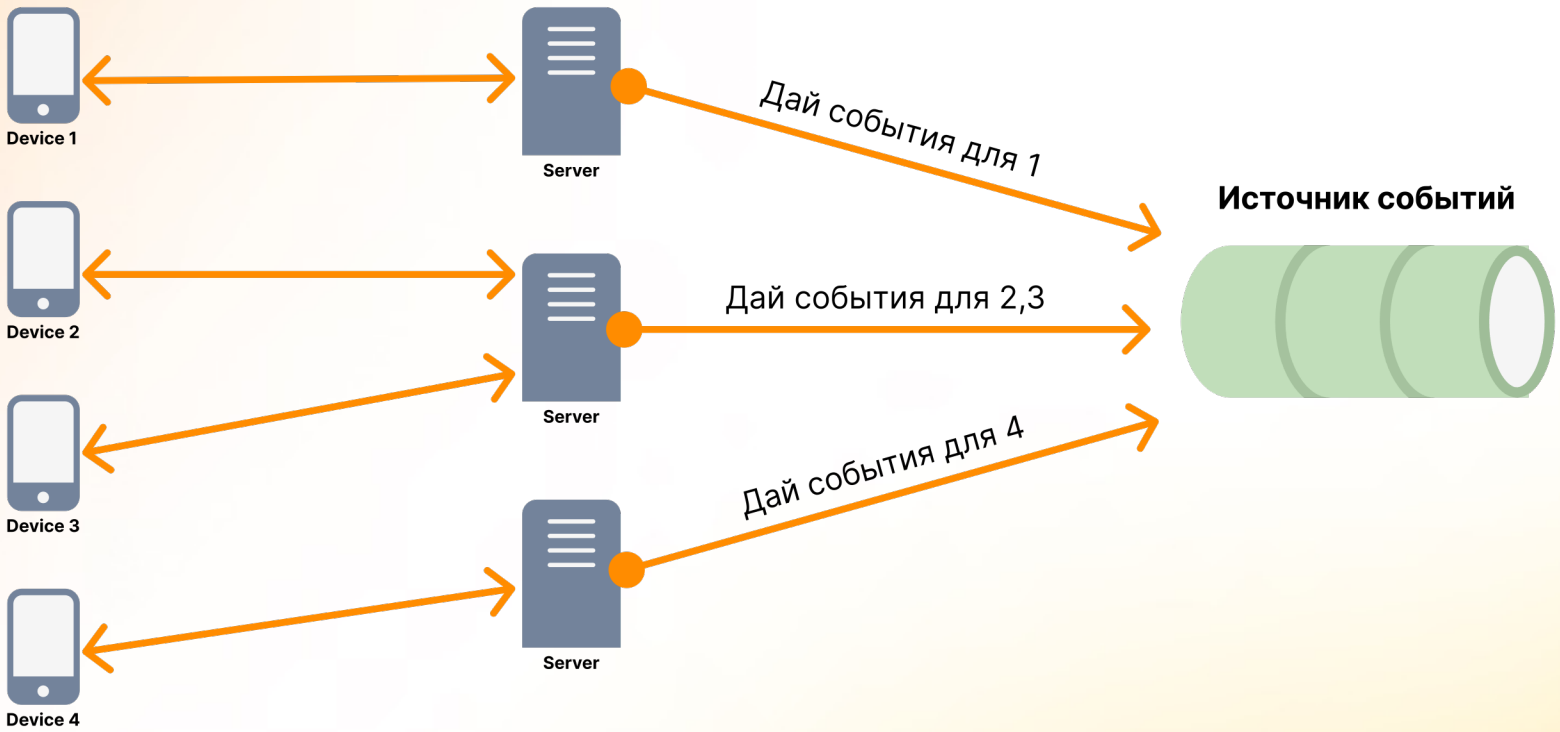


Источник и получатель событий
не знают, какие девайсы подключены

Правило № 3

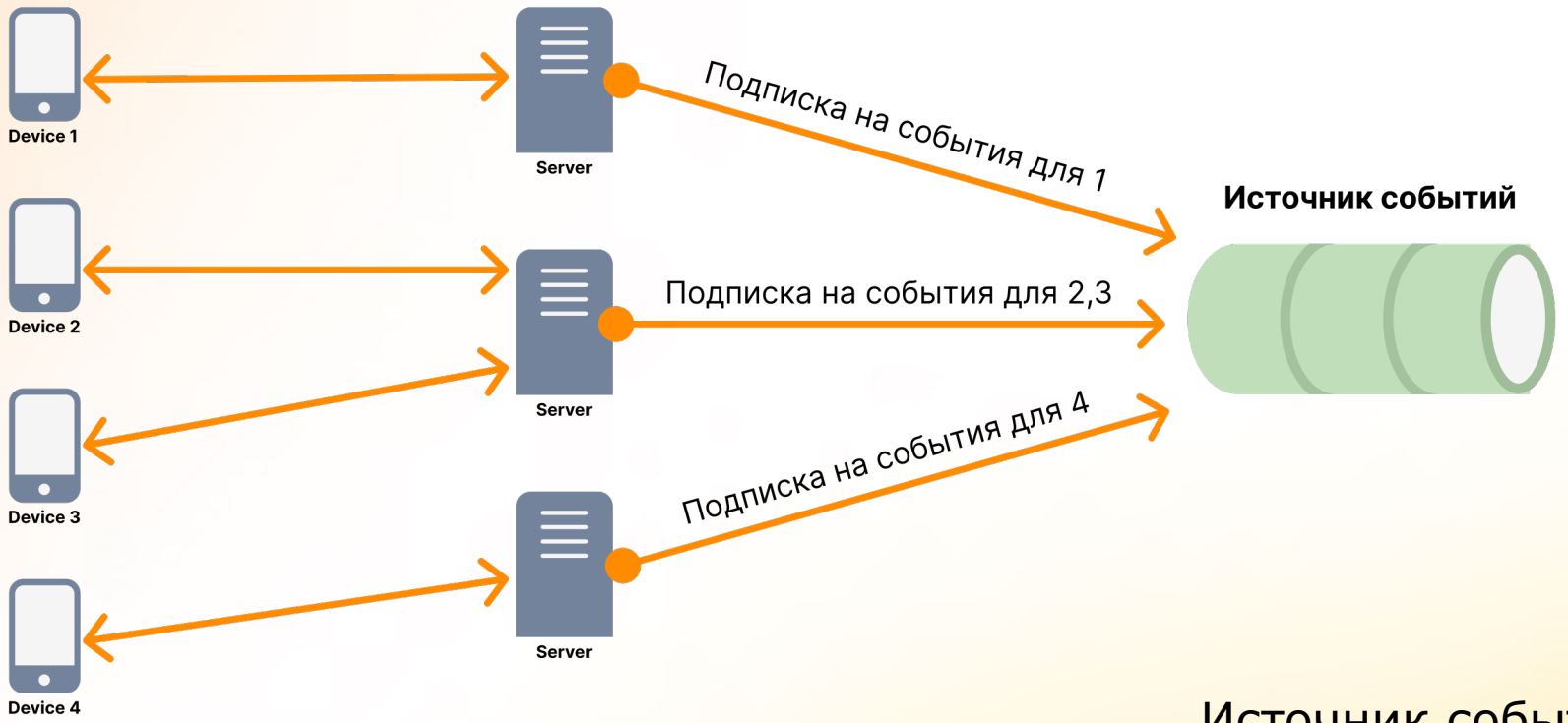
Если вы хотите
масштабировать, заложите
решение в архитектуру заранее

Pull-модель



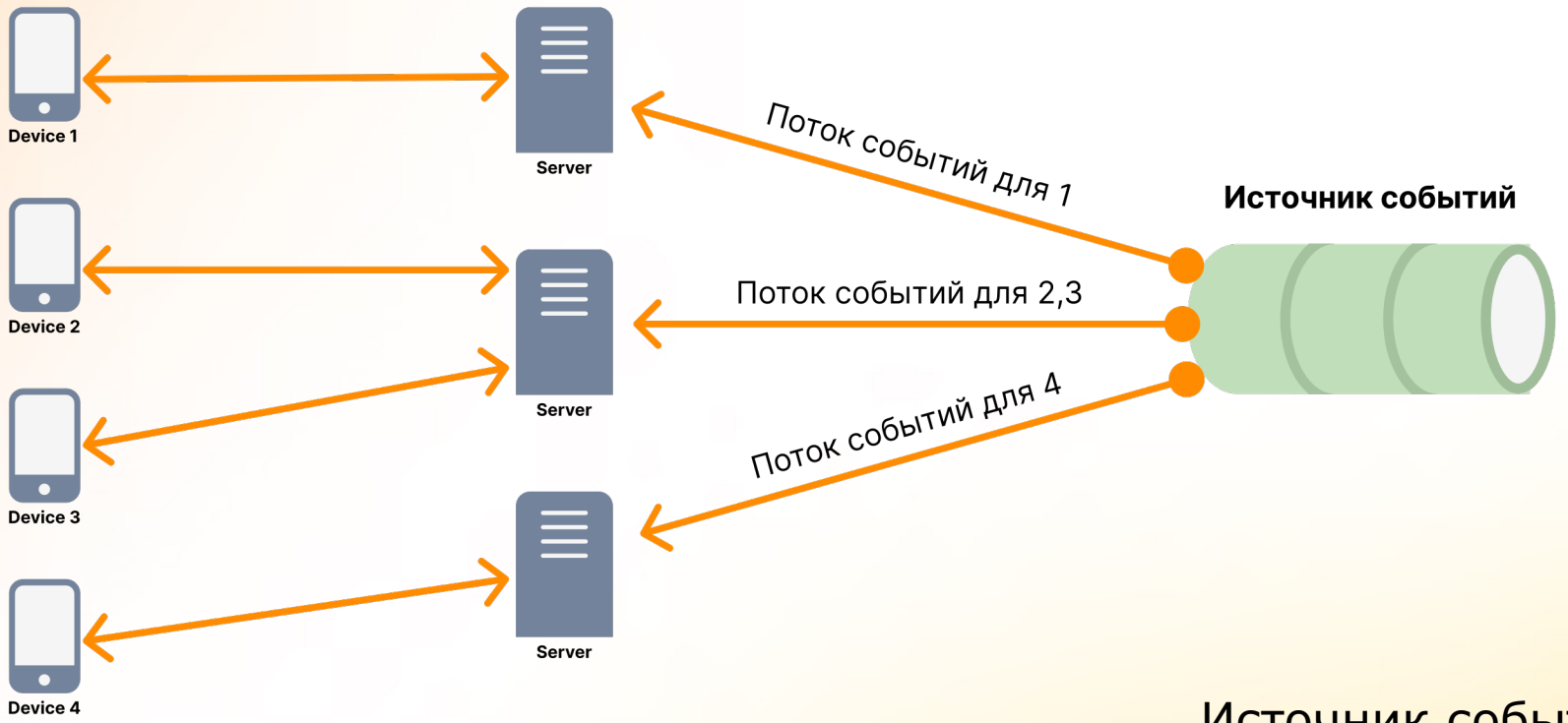
Источник событий - микросервис или БД

Push-модель, шаг 1



Источник событий - очередь или Pub/Sub

Push-модель, шаг 2



Источник событий - очередь или Pub/Sub

Аутентификация

Аутентификация

Создание объекта WebSocket

```
WebSocket WebSocket(
  in DOMString url,
  in optional DOMString protocols
);

WebSocket WebSocket(
  in DOMString url,
  in optional DOMString[] protocols
);
```

Проверьте сами

https://developer.mozilla.org/ru/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

Аутентификация: что бывает

а. ws://user:password@example.org/ws

б. ws://example.org/ws?authorization=<JWT>

в. ws://example.org/ws => auth в первом сообщении

г. ws://example.org/ws?ticket=<some_ticket>

Аутентификация: что бывает

ws://user:password@example.org/ws

ws://example.org/ws?authorization=<JWT>

ws://example.org/ws => auth в первом сообщении

ws://example.org/ws?ticket=<some_ticket>

Правило № 4

Защищайтесь от утечки в
логах критичных данных

О чем не думают бэкендеры*

*поначалу

Сети ненадёжны

Ping-pong

5.5.2. Ping

The Ping frame contains an opcode of 0x9.

A Ping frame MAY include "Application data".

Upon receipt of a Ping frame, an endpoint MUST send a Pong frame in response, unless it already received a Close frame. It SHOULD respond with Pong frame as soon as is practical. Pong frames are discussed in Section 5.5.3.

An endpoint MAY send a Ping frame any time after the connection is established and before the connection is closed.

NOTE: A Ping frame may serve either as a keepalive or as a means to verify that the remote endpoint is still responsive.

5.5.3. Pong

The Pong frame contains an opcode of 0xA.

Section 5.5.2 details requirements that apply to both Ping and Pong frames.

A Pong frame sent in response to a Ping frame must have identical "Application data" as found in the message body of the Ping frame being replied to.

If an endpoint receives a Ping frame and has not yet sent Pong frame(s) in response to previous Ping frame(s), the endpoint MAY elect to send a Pong frame for only the most recently processed Ping frame.

Но реализация очень клиентоспецифична

https://datatracker.ietf.org/doc/html/rfc6455

Что мы обнаружили

1. Chromium не отправляет ping, а в лучшем случае отвечает pong (если сервер делает ping)
2. Клиентские приложения имеют шанс узнать о дисконнекте через 5, 10, 15 минут

Comment 12 by ricea@chromium.org on Tue, May 2, 2017, 5:23 PM GMT+3 **Project Member**

Owner: ricea@chromium.org
Labels: -Pri-2 OS-Mac Pri-3

I just tried this on OS X 10.12.4 and it did eventually disconnect after 18 minutes. I need to do more investigation. Temporarily assigning to me.

https://bugs.chromium.org/p/chromium/issues/detail?id=706002

Очевидное решение: придумать свою интерпретацию стандарта

Data	Length	Time
⬇ a["{\"msg\":\"ping\"}"]	23	19:08:00.274
⬆ ["{\"msg\":\"pong\"}"]	22	19:08:00.274
⬆ ["{\"msg\":\"ping\"}"]	22	19:08:22.896
⬇ a["{\"msg\":\"pong\"}"]	23	19:08:22.904
⬇ a["{\"msg\":\"ping\"}"]	23	19:08:45.275
⬆ ["{\"msg\":\"pong\"}"]	22	19:08:45.295
⬇ a["{\"msg\":\"ping\"}"]	23	19:09:15.277
⬆ ["{\"msg\":\"pong\"}"]	22	19:09:15.298
⬆ ["{\"msg\":\"ping\"}"]	22	19:09:32.872
⬇ a["{\"msg\":\"pong\"}"]	23	19:09:32.894
⬇ a["{\"msg\":\"ping\"}"]	23	19:10:00.309
⬆ ["{\"msg\":\"pong\"}"]	22	19:10:00.309

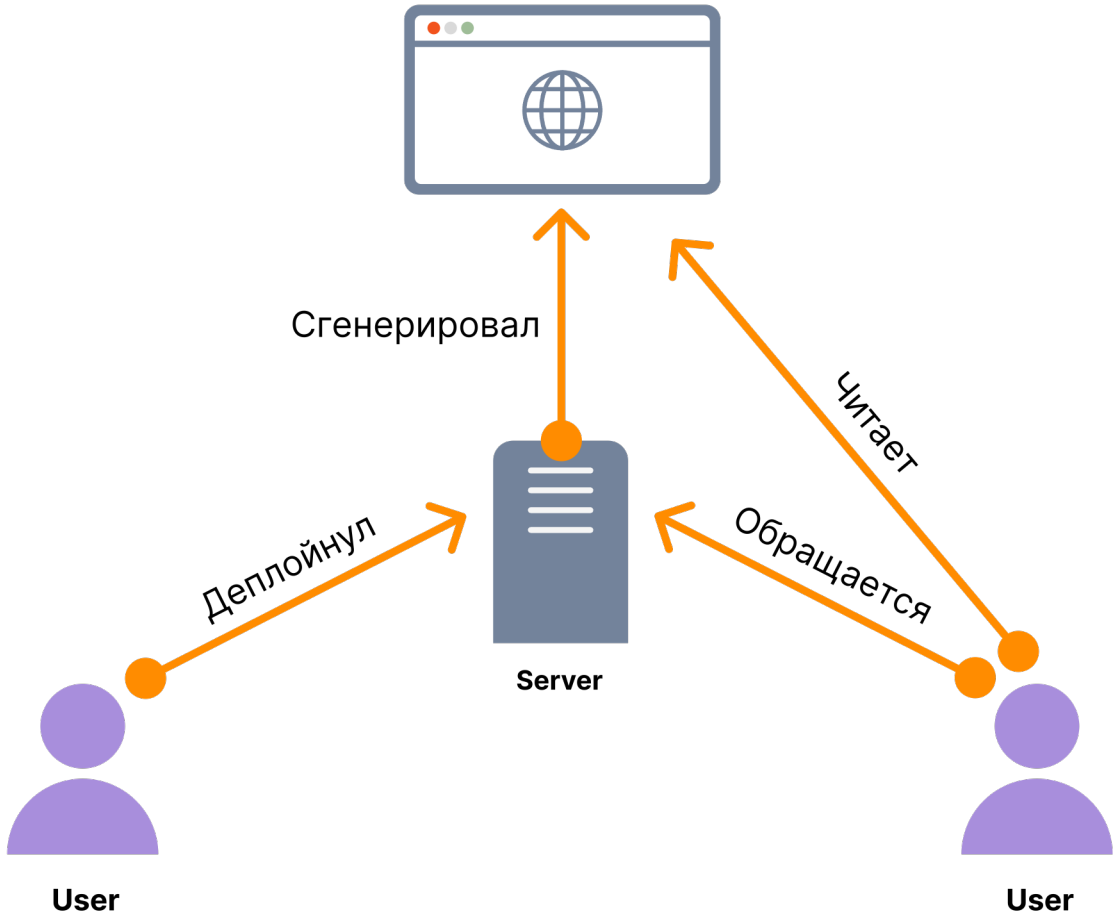
Data	Length	Time
⬇ (ʘ‿ʘ)	5	19:03:13.350
⬇ (ʘ‿ʘ)	5	19:03:33.351
⬇ (ʘ‿ʘ)	5	19:03:53.355
⬇ (ʘ‿ʘ)	5	19:04:13.356
⬇ (ʘ‿ʘ)	5	19:04:33.358
⬇ (ʘ‿ʘ)	5	19:04:53.369
⬇ (ʘ‿ʘ)	5	19:05:13.362
⬇ (ʘ‿ʘ)	5	19:05:33.364
⬇ (ʘ‿ʘ)	5	19:05:53.366
⬇ (ʘ‿ʘ)	5	19:06:13.368
⬇ (ʘ‿ʘ)	5	19:06:33.372
⬇ (ʘ‿ʘ)	5	19:06:53.427

Правило № 5

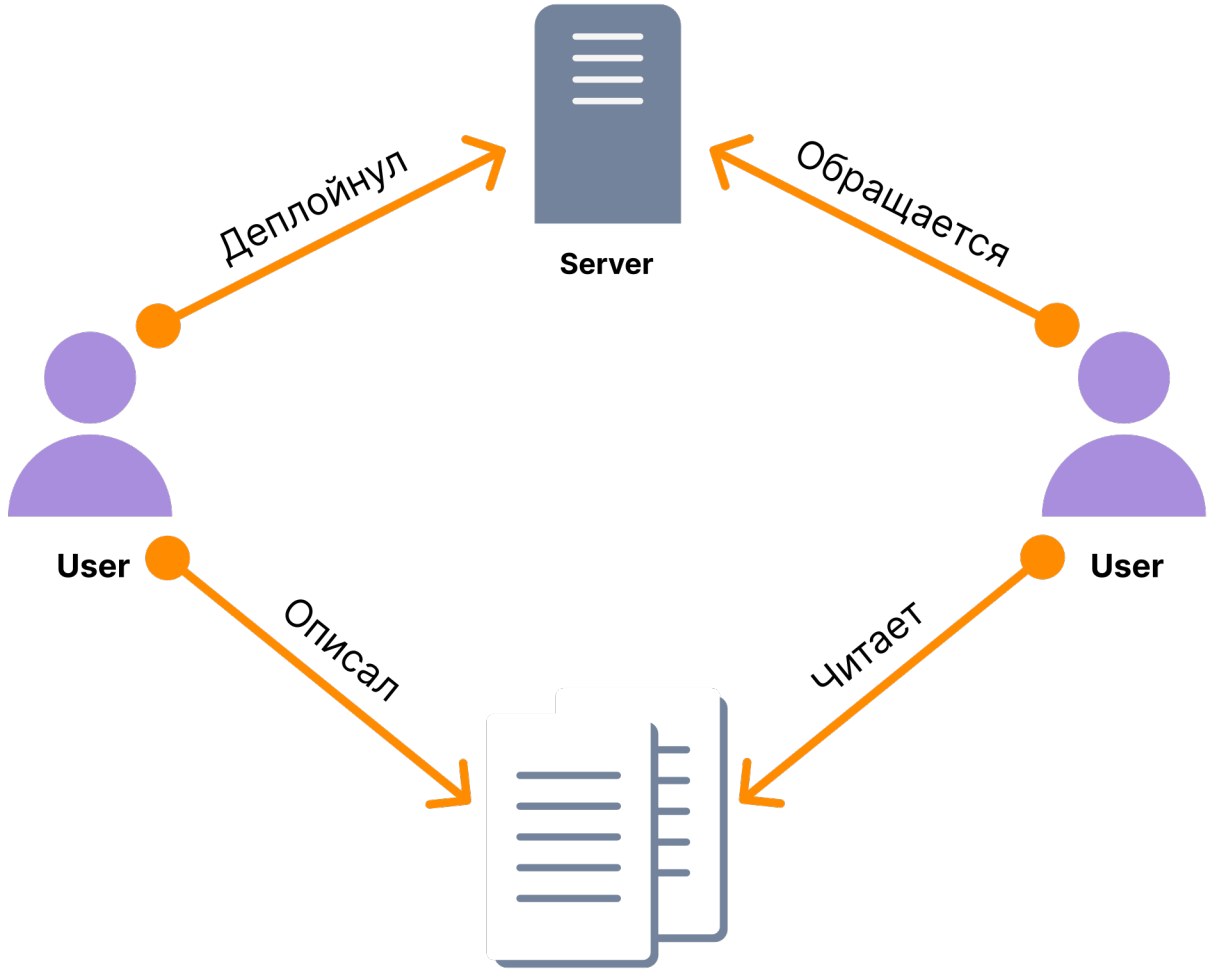
Если стандарт покрывает не все проблемы, докручивайте его под свои нужды

Как жить без OpenAPI

Документация для HTTP

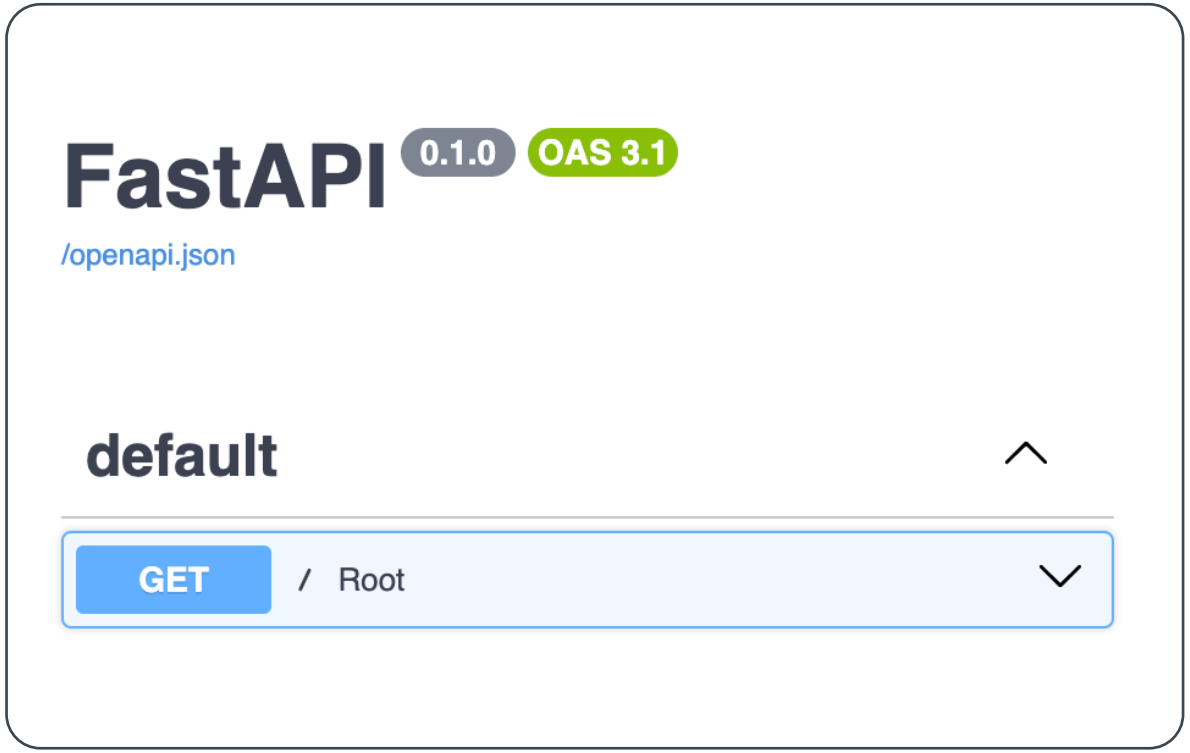


Документация для WebSockets

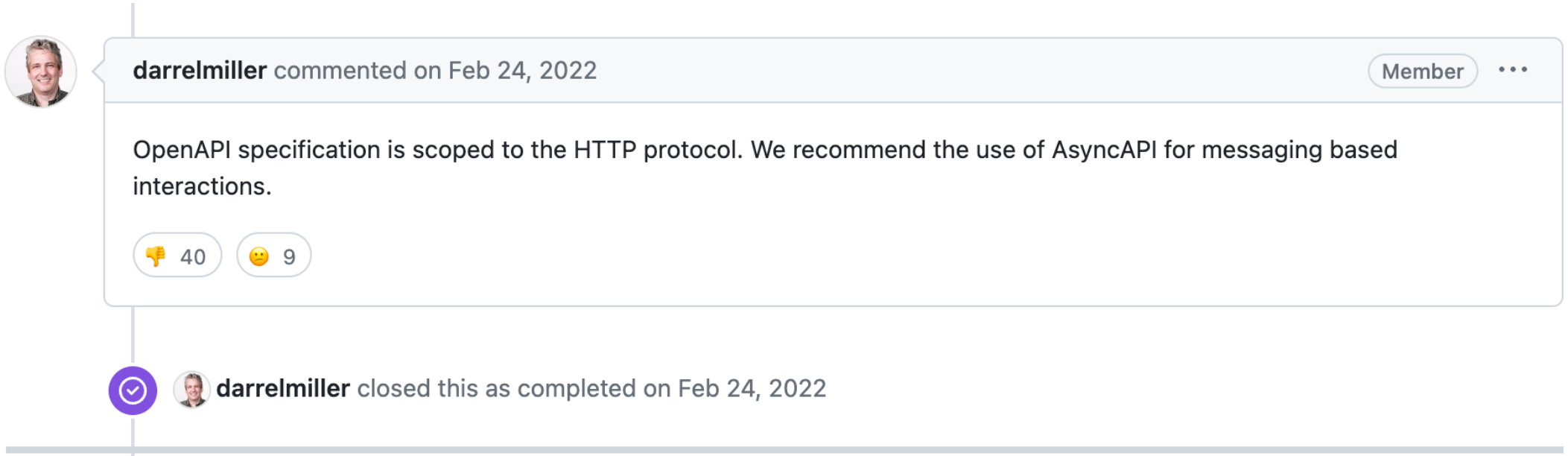


Как жить без openAPI

```python
from fastapi import FastAPI, WebSocket
from fastapi.responses import PlainTextResponse

app = FastAPI()

@app.get("/")
async def root():
    return PlainTextResponse("Hello world")

@app.websocket("/ws")
async def websocket_route(websocket: WebSocket):
    await websocket.accept()
    while True:
        data = await websocket.receive_text()
        await websocket.send_text(f"Message text was: {data}")
```



Как жить без openAPI



https://github.com/OAI/OpenAPI-Specification/issues/55

Как жить без openAPI

1. Define proto

```
import "person_info.proto";

package persons;

message Person {
    PersonInfo info = 1; // characteristics of the person
    repeated Friend friends = 2; // friends of the person
}
```

2. Compile proto

```
_PERSON = _descriptor.Descriptor(
  name='Person',
  full_name='persons.Person',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  create_key=_descriptor._internal_create_key,
  fields=[
...
```

3. Serialize

```
info {
  age: 30
  height: 184
}
friends {
  friendship_duration: 365.1000061035156
  shared_hobbies: "books"
  shared_hobbies: "daydreaming"
  shared_hobbies: "unicorns"
  person {
    info {
      age: 40
      height: 165
    }
  }
}
```

Правило № 6

Contract First

Выводы

1. Следите за согласованностью клиентского состояния
2. Не используйте Websocket для обмена событиями, которым требуется подтверждение обработки
3. Если вы хотите масштабировать, заложите решение в архитектуру заранее
4. Защищайтесь от утечки в логах критичных данных
5. Если стандарт покрывает не все проблемы, докручивайте его под свои нужды
6. Contract First

Михаил Житков

Работаю в QIWI, команда Qchat

LifeforDream

Вопросы