github: https://github.com/nikvas0/compression-project

# Twitch-like

## Про протоколы и кодирование/декодирование

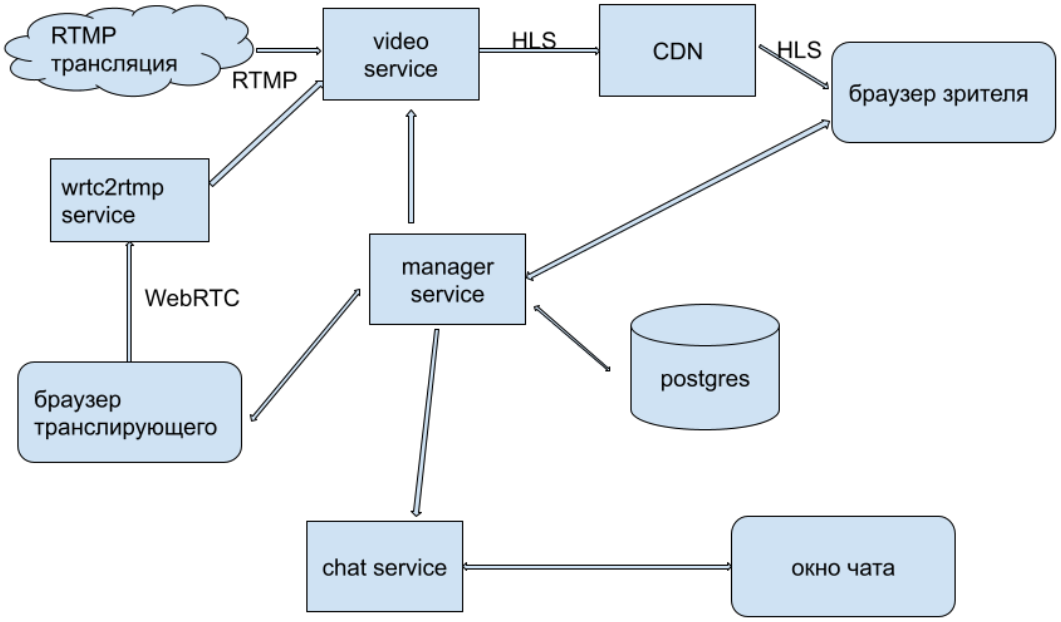
Твич использует RTMP для отправки видео и HLS streams для трансляции. Будем делать все также, так как HLS поддерживается браузерами нативно (имеет поддержку nginx + просто работать с CDN), используем его.
Сервис будет принимать RTMP поток, затем перекодировать через ffmpeg (несколько разных битрейтов сразу, чтобы его можно было выбирать) и раздавать трансляцию через nginx.
Так как понадобится еще клиент, который будет сайтом, то добавляется еще сервис, получающий данные через WebRTC и перекодирующий в RTMP.

## Интерфейс

У веб-клиента будет только 3 страницы:

- Главная со списком трансляций;
- Просмотр трансляции с видео и чатом;
- Ведение трансляции с видео и чатом.

# Архитектура




manager service -- сервис, предоставляющий REST API для управления стримами.
video service -- сервис, преобразующий RTMP в HLS используя ffmpeg.
wrtc2rtmp service -- прием видео по WebRTC и преобразование в RTMP (pion WebRTC + ffmpeg).
chat service -- сервис чатов.

# API

manager service

## /streams

Выдается список имен стримов со ссылками на них.

## /start_stream?name=...

Для стрима генерируется id публичный и приватный, после чего manager service отправляет id в video service и chat service команду создать новый

стрим и чат-комнату, они в свою очередь возвращают адрес комнаты и адреса стрима. Затем сохраняет все полученное в базе и возвращает клиенту.

### /stream/<public id>?bitrate=...

Находит по публичному адресу ссылку на просмотр стрима (HLS) с нужными характеристиками (битрейт) и ссылку на чат, возвращает их.

### /stop_stream/<public id>, в хедерах: <private id>

Удаляет стрим из базы, останавливает его в video service и chat service.

wrtc2rtmp service

### /start_wrtc_stream/<public id>, в хедерах: <private id>

Получает из базы адрес трансляции rtmp и сохраняет его локально. Должно быть выполнено до начала трансляции по WebRTC.

## Стадии реализации

1. video service + manager service
2. просмотр трансляции через hls
3. трансляция + wrt2rtmp service
4. чат весь
5. CDN

## Замечания

Как видно все сервисы здесь легко масштабируются (postgres можно пошардировать при надобности), но при падении сервиса video/wrtc2rtmp/chat service сломается соответствующая функциональность у стрима. Можно следить за доступностью сервисов и повторять логику /new_stream без перегенерации id, однако этого мы делать не будем.

wrtc2rtmp возможно будет переделан так, чтобы сразу транслировать HLS, однако это сопряжено с рядом проблем с работой на устройствах Apple([https://flashphoner.com/do-not-broadcast-webrtc-streams-in-hls-or-the-first-connected-viewer-issue/](https://flashphoner.com/do-not-broadcast-webrtc-streams-in-hls-or-the-first-connected-viewer-issue/)). Также это потребует от сервиса принимать команды от manager, что еще больше добавит работы.

## Ссылки

https://github.com/pion/webrtc

https://blog.twitch.tv/en/2015/12/18/twitch-engineering-an-introduction-and-overview-a23917b71a25/
https://blog.twitch.tv/en/2017/10/10/live-video-transmuxing-transcoding-f-fmpeg-vs-twitch-transcoder-part-i-489c1c125f28/
https://github.com/pion/example-webrtc-applications/tree/master/twitch
https://docs.peer5.com/guides/production-ready-hls-vod/
https://habr.com/ru/post/401107/
https://github.com/gwuhaolin/livego
https://developer.mozilla.org/en-US/docs/Web/Guide/Audio_and_video_delivery/Live_streaming_web_audio_and_video
https://flashphoner.com/do-not-broadcast-webrtc-streams-in-hls-or-the-first-connected-viewer-issue/