

Вопросы и требования:

1) Может ли система масштабироваться?

Ответ: да может потому что есть координирующий сервер который обеспечивает удобный и простой интерфейс получения доступа ко всем серверам ЖК

2) Может ли у одного ЖК быть несколько копий и как это работает?

Ответ: да, может потому что координатор получает список всех серверов ЖК и при создании ЖК

координатор получает свободный сервер или если происходит подключение с помощью кода координатор опрашивает

сервера для того чтобы узнать какому серверу принадлежит код и происходит подключение

3) Откуда координатор получает данные о всех ЖК и данные о запущенных серверах?

а) на первом этапе координатор будет получать данные о доступных серверах из JSON файла

б) на втором этапе координатор будет получать данные из удалённой базы данных MySQL

4) Как работает и где хранится информация о планировании?

Ответ: планирование происходит на сайте для конкретного ЖК через дату и время, определяем можем ли мы

выбрать дату и время на основе количества серверов ЖК для данного ЖК и уже имеющихся записей через координатор,

данные хранятся в соответствии с пунктом 3, если пользователь хочет задержаться в сессии когда время

истекает можно предложить продлить сессию если свободно время

5) Какие проблемы с серым ip?

а) В данный момент проблема не актуальна, но есть решение проблемы если сервера жк будут сами подключаться к координатору

б) но остаётся неопределённая проблема с WebRTC и серым ip

7) Можно ли завершить сессию если все пользователи сессии неактивны?

если все пользователи сессии неактивны в течение N времени то сессия завершается

Краткое описание архитектуры стриминга:

старый способ начать сессию:

демонстрация->выбор_жк (переходим на новый сайт)
->начать_демонстрацию->начать_новую->получить_код_&_соединиться->начать (переходим на новый сайт)

6 пунктов и 3 сайта

новый способ начать сессию:

демонстрация->выбор_жк->запланировать_новую ИЛИ начать_новую ИЛИ
подключиться_к_существующей

3 пункта и 1 сайт

1) подключаемся к координатору сразу как только заходим на сайт или нажимаем кнопку демонстрации

2) после нажатия на кнопку демонстрации получаем список серверов у координатора и отображаем список возможных ЖК

3) при нажатии на ЖК, появляется возможность создать , подключиться или запланировать сессию

4)

а) если создали сессию подключаемся к ней сразу же, посылаем координатору запрос на создание сессии определённого жк,

он опрашивает сервера для получения свободного, если найден свободный сервер посылает запрос на создание сессии и происходит подключение к серверу ЖК,

если свободный сервер не найден возвращает ошибку

b) если хотим подключиться то вводим код и нажимаем подключиться, координатор опрашивает сервера ЖК на знание кода,

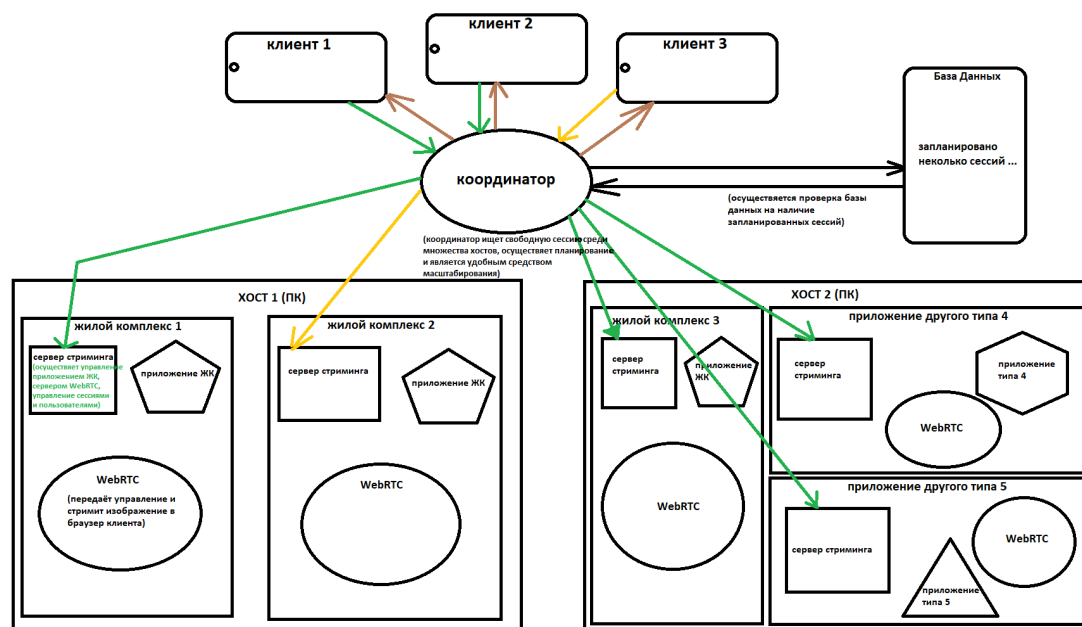
если найден ЖК знающий код то подключаемся к серверу ЖК, если нет то получаем код ошибки

Описание архитектуры координатора:

Задача координирующего сервера связать сайт с сервером стриминга (см pixelServer.txt),
запланировать сессию,

координирующий сервер знает список серверов с помощью файла конфига который загружает
(обновляет) раз в N (30) секунд:

1 этап - установка соединения с сервером стриминга



1) имеет 2 файла конфига servers.json и configuration.json

a) "servers.json" содержит список серверов "массив верхнего уровня": и их название, адрес, описание вида

{

```
"server_name1":"SomeName1",  
  
"ip":"0.0.0.0", "port":"111",  
  
"rc_info":{"rc_name":"SomeRC",  
  
"info":{  
  
    "deskription":[  
  
        {"en":"best RC and ..."},  
  
        {"ru":"лучший ЖК и"}  
  
    ]}  
  
}
```

b) "configuration.json" содержит конфигурацию координатора

```
{"ip":"0.0.0.2", "port":"333", "time_between_updates_seconds":"30"}
```

2) Подключаемся к координатору с помощью websocket и SSL,

отправляем запрос {"verb":"GET_SERVERS", "range":"X-Y"} (range это сервера с X до Y,

например с 0-20 или 20-213, количество серверов = Y - X + 1)

a) в случае успеха получаем {"msg":"SERVERS_FOUND", "servers":{"данные из примера
"1)a)" }}

b) в случае неудачи получаем {"msg":"SERVERS_NOT_FOUND"}

3) Пользователь планирует сессию с помощью запроса {"verb":"PLAN_SESSION",
"date":"day/month/year", "time":"00:00"}

a) координатор сохраняет сессию в список запланированных сессий, файл формата JSON,
sessions_planned.json

и выдаёт четырёхзначный код с помощью которого можно будет подключиться к сессии в
назначенное время {"msg":"CONNECTION_CODE", "code":"1234"}

4) Если пользователь хочет создать сессию без планирования то посылает запрос {"verb":"START_SESSION", "rc_name":"SomeRC"}, координатор создаёт сессию

на свободном сервере и возвращает {"msg":"SESSION_CONNECTION_DATA", "ip":"0.0.0.1", "port":"1111"},

если все сервера заняты вернёт {"msg":"SERVERS_BUSY"},

когда координатор получает запрос "START_SESSION":

a) координатор проверяет список планирования чтобы убедиться что на данное время нет активных записей затем он посылает

каждому серверу в порядке очереди запрос на создание сессии {"verb":"START_SESSION"}, если создать сессию можно

сервер стриминга вернёт {"msg":"SESSION_CONNECTION_DATA", "ip":"0.0.0.1", "port":"1111"}

b) когда клиент получил данные для подключения к сессии он устанавливает соединение с сервером стриминга с помощью сокета

и разрывает соединение с координатором (может быть не стоит разрывать соединение с координатором)

5) Если пользователь хочет подключиться к сессии с помощью кода то посылает запрос

{"verb":"CONNECT_SESSION_WITH_CODE", "code":"0000"}, в ответ придёт {"msg":"SESSION_CONNECTION_DATA", "ip":"0.0.0.1", "port":"1111"}, или

если сессии не существует {"msg":"SESSION_NOT_EXIST"}

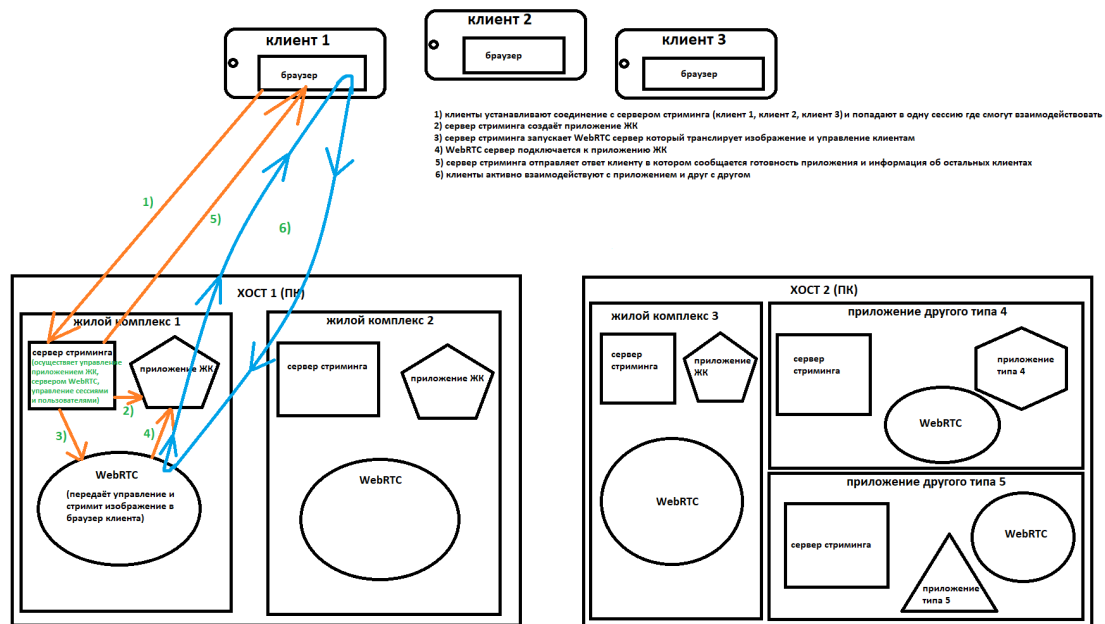
Описание архитектуры сервера сессий:

Задача стримингового сервера создавать сессии по запросу координатора, подключать пользователей к сессии,

запускать приложение ЖК и WebRTC сервер в рамках одной сессии, управление пользователями (идентификация пользователей, присвоение управления),

завершение сессии, приложения ЖК и WebRTC сервера.

2 этап - стриминг приложения в браузер клиента



1) Сервер стриминга имеет файл config.json в котором содержатся данные о (ip, port, допустимое количество сессий и данные для работы WebRTC).

2) Координирующий сервер присылает запрос на создание сессии {"verb":"START_SESSION"}, если лимит сессий не превышен создаёт сессию,

запускает приложение_ЖК и WebRTC сервер, затем отправляет ответ координатору {"msg":"SESSION_CONNECTION_DATA", "ip":"0.0.0.1", "port":"1111"},

если лимит сессий превышен отправляет ответ {"msg":"SESSION_LIMIT_REACHED"}

3) Затем пользователь подключается к сессии с помощью запроса {"verb":"CONNECT_TO_SESSION", "ip":"0.0.0.1", "port":"1111"} и сервер

стриминга отвечает {"msg":"SESSION_CONNECTION_ESTABLISHED", "WebRTC_connection_data":"https://ip:port"} (список данных дополнится для клиента),

после чего пользователь подключается к стриму (процесс подключения дополнится для клиента)

4) Во время соединения пользователи могут обмениваться данными через сессию

5) Если все пользователи отключились от сессии то закрывается приложение ЖК, сервер WebRTC и уничтожается сессия