

# 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 сессий

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

Ответ: да, может потому что координатор получает список всех серверов сессий 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 сессий?

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 сессий 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) Как происходит авторизация и ограничение планирования/создания сессий одним пользователем?

а) Предусмотрена регистрация/авторизация, после чего администратор производит модерацию и принимает/отклоняет запрос

на создание аккаунта который может создавать и планировать сессии (с определёнными ограничениями, например ограничение

количества сессий которые можно создать в определённый временной период)

b) После чего подключиться к сессии можно будет с помощью кода который получил пользователь при планировании сессии

c) Пользователям которые не планировали или не создавали сессию можно будет подключаться без авторизации

6) Как будет решена проблема с серым ip на серверах сессий?

a) Все сервера сессий будут общаться с клиентом через координатор

b) Все сервера WebRTC будут общаться с клиентом через STUN сервер

c) потенциально можно использовать архитектуру  
клиент->координатор->WebRTC->приложение

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 || начать\_новую || 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 сессию или подключиться

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

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

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

## Архитектура координатора:

Задача координирующего сервера создать сессию для клиента или связать клиента с сессией, запланировать сессию,

координирующий сервер знает все активные сессии и список серверов сессий с помощью файла конфига servers.json (обновляет раз в N секунд)

и постоянного соединения с серверами:

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

а) "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,

a) при подключении к координатору пользователь получает уникальный id

```
{"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", "rc\_name":"SomeRC", "date":"day/month/year", "time":"00:00"}

a) в браузере доступен выбор только того времени которое не занято

b) 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":"CREATE\_SESSION", "rc\_name":"SomeRC"},

координатор создаёт сессию на свободном сервере и возвращает {"verb":"CONNECT\_TO\_SESSION", "ip":"0.0.0.1", "port":"1111", "code":"1234"},

если все сервера заняты вернёт {"msg":"SERVERS\_BUSY"}, когда координатор получает запрос "CREATE\_SESSION":

a) координатор проверяет список планирования чтобы убедиться что на данное время нет активных записей, затем свободному серверу сессий

отправляет запрос на создание сессии {"verb":"CREATE\_SESSION", "rc\_name":"SomeRC", "user\_id":"id123"},

если создать сессию можно, сервер сессий вернёт {"verb":"CONNECT\_TO\_SESSION", "ip":"0.0.0.1", "port":"1111", "code":"1234"}

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

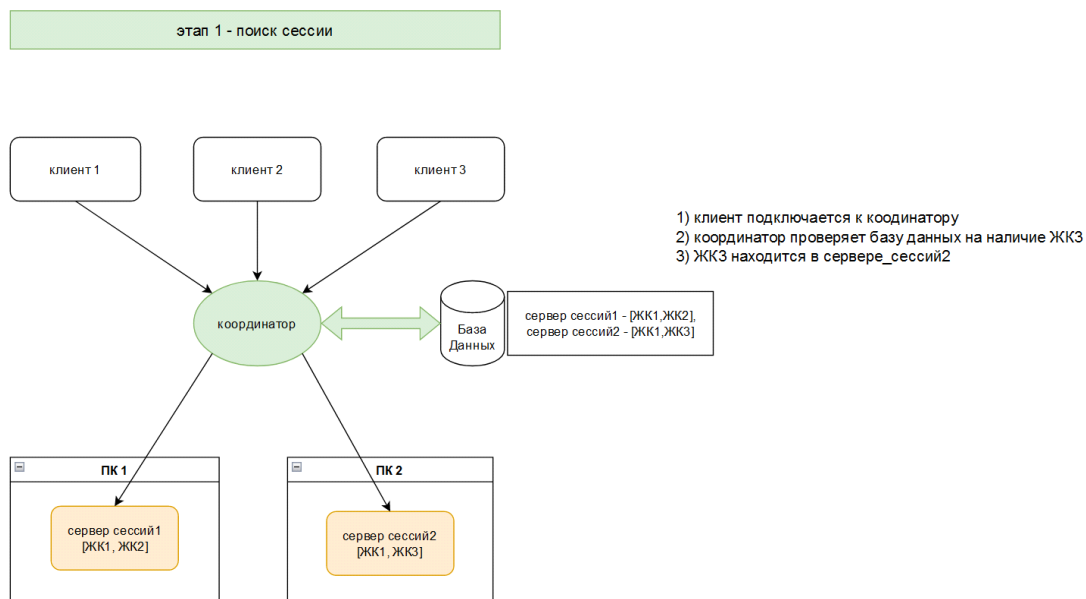
и разрывает соединение с координатором (может быть не стоит разрывать соединение с координатором)

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

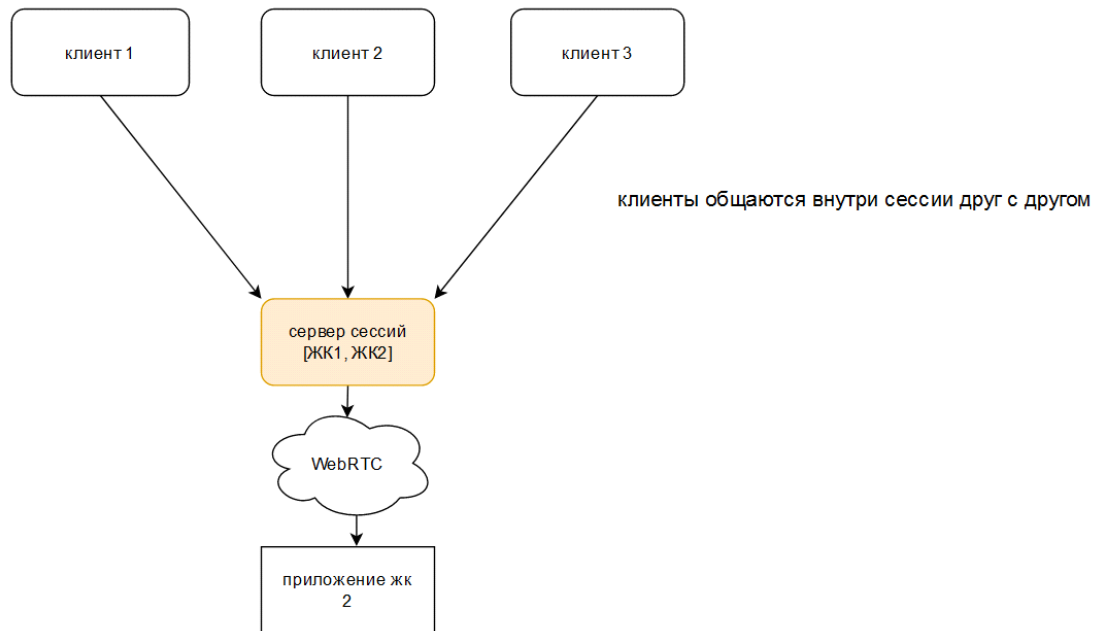
{"verb":"CONNECT\_SESSION\_WITH\_CODE", "code":"1234"},

в ответ придёт {"verb":"CONNECT\_TO\_SESSION", "ip":"0.0.0.1", "port":"1111", "code":"1234"},  
или

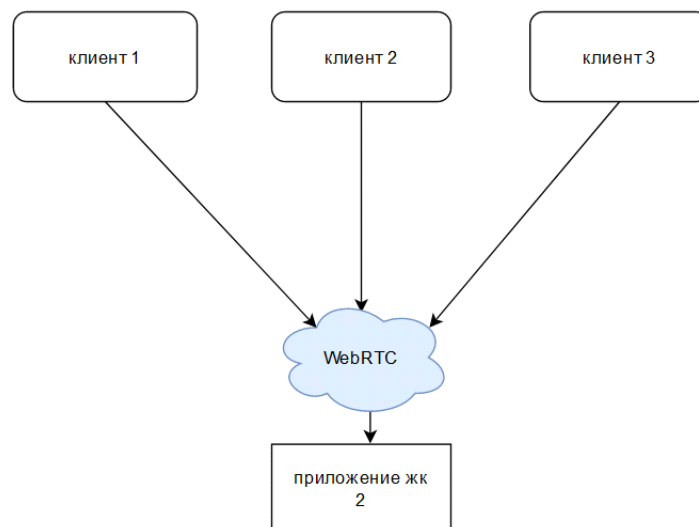
если сессии не существует {"msg":"SESSION\_NOT\_EXIST"}.



этап 2 - подключение к сессии и WebRTC серверу



потенциальный этап 3 - подключение к сессии через WebRTC сервер



Архитектура сервера сессий:

Задача стримингового сервера создавать сессии по запросу координатора, поддерживать соединение с клиентами в рамках сессии,

запускать приложение\_ЖК и WebRTC сервер в рамках одной сессии, управление пользователями (присвоение управления и т.д.),

завершение сессии, приложения\_ЖК и WebRTC сервера.

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

2) Координирующий сервер знает лимит сервера сессий и присылает запрос на создание сессии {"verb":"CREATE\_SESSION", "user\_id":"id123"},

запускает приложение\_ЖК и WebRTC сервер, затем отправляет ответ координатору

{"verb":"CONNECT\_TO\_SESSION", "ip":"0.0.0.1", "port":"1111", "code":"1234"}

3) Затем координатор подключает пользователя к серверу сессий с помощью сокета ("ip":"0.0.0.1", "port":"1111") и присылает запрос

{"verb":"CONNECT\_USER\_TO\_SESSION", "code":"1234", "user\_id":"id123"}, после чего сервер сессий отправляет 2 запроса

первый запрос координатору который говорит о том что пользователь подключился

{"msg":"SESSION\_CONNECTION\_ESTABLISHED", "WebRTC\_connection\_data":"https://ip:port"}  
(список данных дополнится),

после чего пользователь подключается к стриму (процесс подключения дополнится)

4) Во время соединения пользователи могут обмениваться данными внутри сессии (список данных дополнится)

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