Проект выполнен с помощью проекта Draw io. Открыть его можно тут: <https://app.diagrams.net/>

# Взаимодействие клиент-сервер.

В рамках домашнего задания было принято использовать брокер сообщений.

1. Клиент выполнен в виде монолита. Каждый клиент взаимодействует с сервером через очередь **client\_request\_queue**. В ней передаются все необходимые запросы на сервер:
   1. Запрос токена авторизации.
   2. Отправка команд игры.
   3. Получение ссылки на игру и её состояние.
   4. Получение своего места в рейтинговой таблице.

К каждому запросу (кроме запроса на получение токена) должен быть приложен токен игрока.

1. Сервер выполнен в виде наборов микросервисов. При получении запроса на получение токена микросервис **Request resolver** передаёт данные авторизации в микросервис **Auth resolver** с помощью очереди **credentials\_queue.** Для проверки корректных данных авторизации **Auth resolver** получает ответ за запрос в БД **Credentials**. В ответ он получает токен авторизации, который был получен из БД **Tokens** и отдаёт его игроку, если данные авторизации были верны. Если токен отсутствовал, он будет сгенерирован и внесён в данную БД. В дальнейшем игрок будет прикладывать этот токен ко всем запросам.

**Request resolver** взаимодействует с **Auth resolver** для проверки токенов с помощью очереди **tokens\_queue**. Если он корректный, то **Request resolver** отправляет команду на выполнение в **Command resolver** с помощью очереди **commands\_queue.**

В **Command resolver** команда десериализуется и отправляется в микросервис **Request resolver** конкретной игры. Внутри каждой игры реализована очередь **command\_queue** для отправки команды на исполнение и получение состояния игры.

1. **Command resolver** при завершении игры отправляет результаты на сервер через собственный менеджер запросов **Request resolver** в **Command resolver** сервера с помощью очереди **commands\_queue**. **Command resolver** сервера отправляет полученные данные в **Request manager** сервера, после чего отправляется запрос в **Ranks resolver** с помощью очереди **ranks\_queue**. В данном микросервисе происходит десериализация комманды и запись результатов в БД **Ranks.**

# Узкие места

1. **Request resolver.** Вполне вероятно, что будет существенно меняться набор команд, их формат. При изменении любого из микросервисов, вполне возможно, придётся изменять и Request resolver.
2. Доставка команды до игры происходит слишком долго. Для отправки команды необходимо проверить токен, сделав запрос в БД. Возможным решением является хранение в оперативной памяти сервера список токенов для каждой конкретной игры.

При работе клиента и игры напрямую возможны трудности в реализации безопасного взаимодействия с игроков.

1. Набор серверов обслуживания клиентов. При большом количестве игроков и разном их географическом расположении необходимо создать несколько серверов, структура которых была описана выше. Для принятия решения, с каким сервером взаимодействовать, необходимо создать ещё один микросервис. Его задачей будет предоставление наиболее подходящего сервера клиенту.
2. Исходя из решения проблемы выше необходимо настроить синхронизацию с различными БД **Ranks** для получения сведения о своём положении игрока среди всех остальных.

# Наиболее часто изменяемые компоненты.

1. Все реализации **Command resolver’ов**. С развитием игры будет увеличиваться число команд, которые необходимо интерпретировать. Эти микросервисы должны быть расширяемы за счёт увеличения доступных для выполнения команд.
2. **Auth resolver**. Вполне вероятно, что будет меняться способ авторизации, добавится дополнительная информация в токен авторизации.
3. **Ranks resolver**. От турнира к турниру могут меняться правила подсчёта результатов каждой игры.
4. Самым редко изменяемым компонентом должен оставаться **Request resolver**. Он выполняет роль промежуточного звена при передаче сообщений.