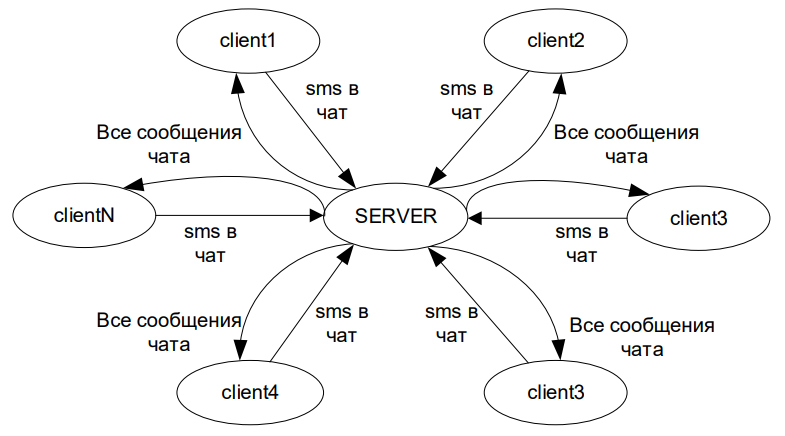
1. **Схема приложений**



1. **Архитектура приложений**

**Клиент**

Как только приложение Клиент запущено, инициируется подключение к серверу. Параметры подключения считываются из файла настроек.

На следующем этапе пользователю предлагается ввести имя для участия в чате. После ввода пользователем имени происходит проверка: нет ли среди участников клиента с таким именем. Список клиентов хранится в сервере. Если имя, введенное пользователем, занято, ему предлагается повторить попытку. Если же имя свободно – клиент становится полноценным участником чата. Теперь он принимает сообщения от сервера, отправляет сообщения в чат, и добавляется в список участников.

Каждый клиент задействует 1 поток с неблокирующим подходом (Non-Blocking IO), чтобы иметь возможность писать новые сообщения в чат и без блокирования обрабатывать сообщения от сервера и записывать их в свой личный, именной файл логирования.

Для выхода из чата клиенту нужно набрать команду “/exit”, после чего взаимодействие с сервером прерывается, и имя клиента стирается из файла с именами участников.

После повторного подключения данного клиента его файл логирования не обнуляется, а продолжает дополняться новыми сообщениями от сервера.

Пока под вопросом. Возможно, будут созданы отдельные классы для работы с потоками ввода-вывода. Для удобства и компактности основного кода.

**Сервер**

Сервер запускается с параметрами, считанными из файла с настройками.

Задача сервера заключается в том, чтобы ретранслировать сообщения от каждого клиента всем участникам чата.

Основной поток сервера:

1. Ловит новые подключения клиентов.
2. Предлагает клиенту ввести имя и проверяет, не занято ли оно. Если в чате уже есть участник с таким именем, клиенту предлагается выбрать другое имя.
3. Когда имя выбрано, в пул потоков, отвечающий за взаимодействие с клиентами, добавляется новый процесс. А коллекция listMember< СhatMember> пополняется новым участником (СhatMember – класс, который будет описан ниже).

Для удобства в приложении Сервер будет создан класс Message, который будет хранить всю информацию о сообщении (от кого пришло, время, когда пришло, длина текста в байтах, текст сообщения).

Также в приложении Сервер будет создан класс СhatMember, имплементирующий интерфейс Runnable. Класс СhatMember, помимо метода run() будет содержать методы:

- Метод run() - будет отвечать за прием сообщений от участника чата (с фиксацией времени прихода сообщения), формирование из него объекта класса Message, и запись этого объекта в потокобезопасную очередь allMessages;

- sendMessage(String message) – метод отправки сообщения клиенту;

- removeClient() – метод исключения участника из чата.

Потокобезопасная коллекция allMessages будет сортироваться по времени прихода сообщения от клиента. Когда коллекция не пустая, будет инициирована рассылка накопленных сообщений клиентам и в лог-файл сервера. Отправленное сообщение будет удалено из коллекции.

Пока под вопросом. Возможно, будут созданы отдельные классы для работы с потоками ввода-вывода. Для удобства и компактности основного кода.

**Протокол обмена сообщениями**

Формат сообщений, которыми будут обмениваться клиенты и сервер:

From: <имя клиента>

Data-length: <длина сообщения в байтах>

Message: \r\n

<тело сообщения> \r\n

\r\n

Сообщения от сервера будут идти от имени “server”.